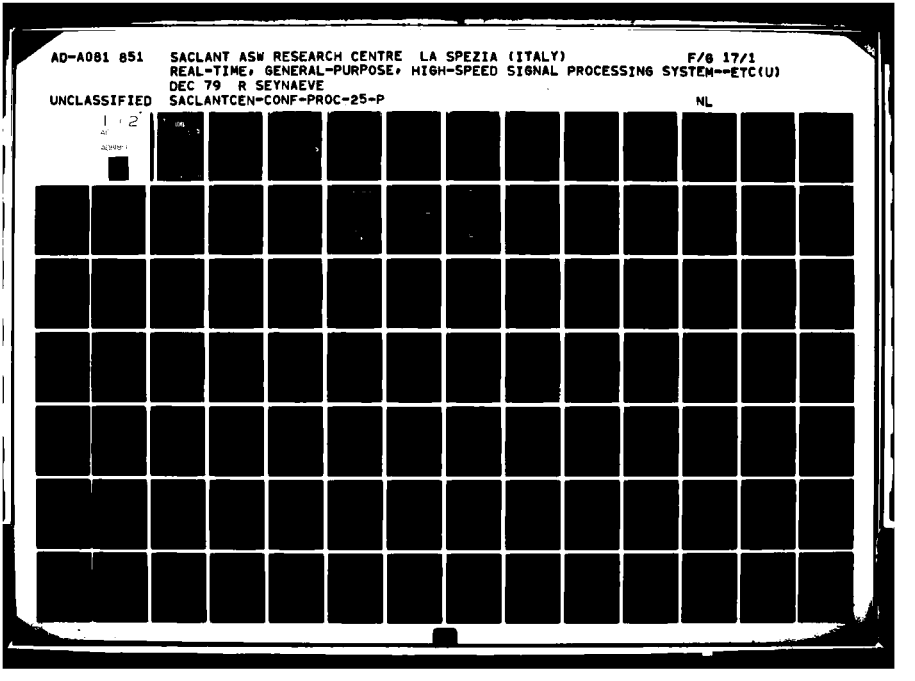

AD-A081 851 SACLANT ASW RESEARCH CENTRE LA SPEZIA (ITALY) F/G 17/1
REAL-TIME, GENERAL-PURPOSE, HIGH-SPEED SIGNAL PROCESSING SYSTEM--ETC(U)
DEC 79 R SEYNAEVE
UNCLASSIFIED SACLANTCEN-CONF-PROC-25-P NL
1 2

Compiled and Published by

STI

(14) SACLANTCEN — CONF-PROC —
(9) CONFERENCE PROCEEDINGS NO. 25-PT-2

NORTH ATLANTIC TREATY ORGANIZATION

SACLANT ASW Research Centre
Viale San Bartolomeo 400, I-19026 San Bartolomeo (SP), Italy.

tel: national 0187 503540
international + 39 187 503540

telex: 271148 SACENT I

A081850

(6)

REAL-TIME, GENERAL-PURPOSE, HIGH-SPEED SIGNAL PROCESSING SYSTEMS FOR UNDERWATER RESEARCH.

Proceedings of a Working Level Conference held at SACLANTCEN on 18-21 September 1979.

(Part 2. SESSIONS IV to VI.)

(10) Organized by
ROBERT SEYNAEVE

(11) 1 December 1979

(12) 138

Supreme Allied Commander's Atlantic Anti-Submarine Warfare Research Center

312950

LIST OF PARTICIPANTS

CANADA

Mr D.V. Crowe
Defence Research Establishment Atlantic
P.O. Box 1012
Dartmouth
Nova Scotia B2Y 3Z7

DENMARK

Mr B. Damsgaard
Danish Defence Research Establishment
Staunings Plads 2
DK 2100 Copenhagen Ø

Mr. P.B. Ring
Danish Defence Research Establishment
Staunings Plads 2
DK 2100 Copenhagen Ø

FRANCE

Mr J.L. Lambla
GERDSM
D.C.A.N Toulon
83800 Toulon Naval

Mr J.Maigre
GERDSM
D.C.A.N. Toulon
83800 Toulon Naval

GERMANY

Dr J.F. Böhme
FGAN-Forschungsinstitut für Hochfrequenzphysik
Königstrasse 2
D-5307 Wachtberg-Werthhoven

Mr H. Herwig
Forschungsanstalt der Bundeswehr für Wasserschall und Geophysik
Klausdorfer Weg 2-24
2300 Kiel

Ing. Kh. Rosenback
FGAN-Forschungsinstitut für Hochfrequenzphysik
Königstrasse 2
D-5307 Wachtberg-Werthhoven

GERMANY (Cont.)

Mr B. Scholz
Forschungsanstalt der Bundeswehr für Wasserschall und Geophysik
Klausdorfer Weg 2-24
2300 Kiel

Dipl.Ing. H. Urban
Fried. Krupp GMBH Krupp Atlas-Elektronik
Postfach 44 85 45
2800 Bremen 44

Dipl.Ing. W. Wagner
Fried. Krupp GMBH Krupp Atlas-Elektronik
Postfach 44 85 45
2800 Bremen 44

ITALY

Prof V. Cappellini
Istituto di Elettronica
Universita di Firenze
Via di S. Marta 3
50139 Firenze

C.C.(AN) D. Nascetti,IN
Marina Militare
Commissione Permanente per gli Esperimenti del Materiale da Guerra
La Spezia 19026

T.V.(AN) G.C. Testino,IN
Marina Militare
Commissione Permanente per gli Esperimenti del Materiale da Guerra
La Spezia 19026

NETHERLANDS

Mr J.G. Schothorst
Physics Laboratory TNO
Oude Waalsdorperweg 63
Post Box 96864
2509 JG Den Haag

Mr C.H. Vlasblom
Physics Laboratory TNO
Oude Waalsdorperweg 63
Post Box 96864
2509 JG Den Haag

NORWAY

Dr H.J. Alker
Electronics Research Laboratory
O.S. Bragstads Plass 4
N-7034 Trondheim-Nth

Mr K. Hansen
SIMRAD A/S
P.O. Box 111
Horten

Dr. Y.C. Lundh
Norwegian Defence Research Establishment
Postboks 25
N-2007 Kjeller

Dr G.B. Pettersen
Norwegian Defence Research Establishment
Postboks 115
N-3191 Horten

UNITED KINGDOM

Dr J.C. Cook
Admiralty Marine Technology Establishment
Queens Road
Teddington, Middlesex

Dr T.E. Curtis
Admiralty Underwater Weapons Establishment
H.M. Naval Base
Portland, Dorset

Prof J.W.R. Griffiths
Department of Electronic & Electrical Engineering
University of Technology
Loughborough
Leicestershire, LE11 3TU

Mr S.F. Meatcher
Admiralty Underwater Weapons Establishment
H.M. Naval Base
Portland, Dorset

Dr D. Nairn
Admiralty Underwater Weapons Establishment
H.M. Naval Base
Portland, Dorset

Dr J.S. Pyett
Admiralty Underwater Weapons Establishment
H.M. Naval Base
Portland, Dorset

UNITED KINGDOM (Cont.)

Mr C. Richardson
Admiralty Underwater Weapons Establishment
H.M. Naval Base
Portland, Dorset

Mr T. Totten
Royal Aircraft Establishment
Farnborough, Hants

UNITED STATES

Prof A.B. Baggeroer
Massachusetts Institute of Technology
Department of Ocean Engineering
Cambridge
Mass. 02139

Mr J.M. Griffin
Naval Research Laboratory
Code 8160
4555 Overlook Avenue SW
Washington D.C. 20375

Mr E. Hug
Naval Underwater Systems Center
New London, Connecticut 06320

Mr F.W. Molino
Naval Underwater Systems Center
New London, Connecticut 06320

Dr B.L. Pennoyer
Naval Ocean Systems Center
San Diego, California 92152

CAPT C.M. Rigsbee. USN
Assistant Commander for Research & Technology
Naval Air Systems Command
Department of the Navy
Washington D.C. 20360

Mr P.A. Rigsbee
Naval Research Laboratory
Code 7594
Washington D.C. 20375

Mr D. Steiger
Naval Research Laboatory
Code 8003
Washington D.C. 20375

Mr J.E. Vernaglia
Naval Underwater Systems Center
New London, Connecticut 06320

UNITED STATES (Cont.)

Mr S. Weinstein
Naval Research Laboratory
Code 7593
Washington D.C. 20375

Mr Y.S. Wu
Naval Research Laboratory
Code 7590
Washington D.C. 20375

SACLANT HQ

CAPT A. Mowing, UKN (Rtd)

SACLANTCEN

Mr P. Bensi
Mr E. Gerwich
Mr V. Monte
Mr F. Jensen
Mr.W. Kuperman
Mr C. Malcheski
Mr M. McCann
Mr P. Mosfield
Mr G. Orsini
Mr R. Seymour
Mr G.C. Vetteri

Other members of SACLANTCEN staff also participated.

TABLE OF CONTENTS

PART 1 - SESSIONS I to III

Table of Contents (Cont'd)

Table of Contents (Cont'd)

USER ORIENTATED SOFTWARE FOR SIGNAL ANALYSIS

by

C Richardson
Admiralty Underwater Weapons Establishment
Portland, Dorset

ABSTRACT A conflict exists between dedicated analysis instruments which are easy to use but inflexible and general purpose computers which are adaptable but require program development. A software package has been developed which permits easy and interactive manipulation of analysis parameters in a general purpose operating system.

INTRODUCTION

The development of signal analysis systems has historically evolved from analogue instruments. With the application of the FFT through the use of a computer, development initially followed with instrument-like equipment, modern examples of which are highly sophisticated. Following the early development it was soon recognised the computer could be used for more specialised tasks, and hence programming languages, suitable for signal analysis, were developed. A conflict was thus introduced; whether an easy to operate instrument with limited functions is preferable to a programmable system offering considerable flexibility but requiring programming skills. Such has been the advance in micro-electronics that hybrid systems are feasible, using multiple processors for different requirements. Indeed such a solution is ideal when the programming needs are for post-processing of standard analysis functions.

A further question is now raised; whether a programmable system is needed for specialised analysis techniques, or more conventional forms of data processing. For example, application of the Cepstrum function or multidimensional transforms requires special application of analysis techniques; whereas automatic sensor correction or narrow-band classification requires conventional data processing.

In the authors experience a general purpose computer operating system with choice of programming languages and signal analysis library has the advantage of unlimited scope of application and widespread software support, this latter factor being frequently underestimated. However, the choice is dictated by the requirements of the research programme, its resources and perhaps personal inclination.

One other essential requirement of any programmable system, used in a general purpose signal analysis environment, is for some form of interactive communication with the software.

A program may be run repeatedly with only minor modifications to the parameter menu which controls its activity. In fact it was the desire to achieve the best of both options, that is instrument-like manipulation with software flexibility, that inspired the soft-panel approach, to be described.

The usefulness of a dedicated instrument can be extended by means of overlay panels which redefine the function of the various controls. Initially the soft-panel concept was simply to mimic a complete control panel on a computer display, using perhaps a light pen to control it. However, a somewhat modified version has been produced, which is conceptually in the form of a book, where the chapters relate to major topics and the pages to detailed aspects. The technique amounts to nothing more than a simplified means of passing control data to a program.

Naturally the work was undertaken in order to achieve the research objectives in hand and consequently it may be considered as under developed. However, the software is felt to be reasonably general and of interest to others requiring such facilities. FORTRAN has been used where possible.

METHOD OF USE

Once the soft-panel program has been initiated, a chapter and page is selected for display, fig 1 shows a typical example. The format was chosen to suit the Teletronix 4010 storage display and consists of a system message area and a status area at the top of the screen, a parameter value area to the right of the screen and a parameter information area occupying the remainder of the display. The latter consists of up to 28 lines of text which may simply be operator information or may describe associated parameters or commands to the computer. In fact each of the 28 lines is unique and is designated as one of 6 different types as follows:-

TYPE 0 The line contains operator information only; no other action is taken.

TYPE 1 The line describes a command to the computer to perform some task.

TYPE 2 The line describes a name parameter, that is an alphanumeric string of up to 12 characters.

TYPE 3 The line describes a two-position switch which can be set off or on. Labels are attached to the switch describing its functions.

TYPE 4 The line describes a decimal parameter the current value of which is given in the parameter area.

TYPE 5 The line describes an integer parameter the current value of which is given in the parameter area.

Having presented a page on the display the program enters 'line mode' whence any reference to lines of type 2, 4, 5 (by typing the line number) invites a new value of the relevant parameter. A reference to a type 3 line automatically changes the switch and its new label displayed, a reference to type 1 activates that command (ultimate completion of which returns automatically to the page), and a reference to type 0 has no effect at all. Since a storage type display is used, new parameter values are entered in the space provided (see fig 1). Normally one or a few parameters would be modified and a command issued which results in some graphical output, or the like. However, the software automatically assess when the display must be refreshed. Parameter values are (invisibly) tagged with preset values and minimum and maximum limits (if desired) and certain keywords are recognised for special treatment, such as activating the various graphics peripherals or resetting the page to its preset conditions. Mistakes or other unacceptable responses are reported in the system message area, which can also be used to report the outcome of an activity.

Using this system the usual repetative sequence of signal analysis operations is much simplified. An operator can easily change the form of side lobe control he is using, or the resolution, or averaging etc and quickly re-run the data. Fig 2 shows a narrow-band display page from which detailed display routines can be initiated. Here, the initial analysis has been completed (with preliminary graphical output) and the results stored by the computer. Actually the programs attempt to find the requested function from whatever result file is presented, but TRI-SPECTRUM results (ie two auto spectra and the cross spectrum) can be used to derive any display format (1). Furthermore, certain chapters may be devoted to general operating instructions, or details of analysis procedures etc, thus permitting a teach-yourself capability. Other embelishments, which have not been described, extend the usefulness and ease of operation of the system.

IMPLEMENTATION

The principle of operation is straightforward. Each page is stored on magnetic disk in a direct access file. The soft-panel program references the text information as well as the parameters values, which may get modified; it arranges this information on the display and proceeds interactively with the user. A command line terminates this program and initiates the user program which in turn re-activates the soft-panel program on completion.

A library of sub-routines is available to the user permitting programs to be developed and incorporated into the facility without difficulty. In particular parameter values are found by a sub-routine whose parameter list is variable, so any number of parameters can be obtained from any position on the page. Having obtained the input data in this way, or any other way (the data being analysed will be obtained from another source) there are no restrictions on the complexity of the program.

FINAL REMARKS

The object of this presentation has been to illustrate some ideas on the use of computers in research involving signal analysis, consequently details of the programming have been avoided. Also the resources are not available to develop foolproof software, or a system which is as good as the imagination can make it.

Use of a storage type display, rather than a video one, imposes limitations on what can be achieved. On the other hand of course, this type of display is a low cost device with low software overheads and good resolution.

The author has used this system as a building block for chapters on routine third octave analysis, narrow-band analysis, multidimentional analysis as well as certain mathematical models in structural acoustics.

REFERENCES

1. Otnes, RK and Enochson, L
Digital Time Series Analysis
Wiley

DISCUSSION

R. Seynaeve What sort of users do you have?

C Richardson As a small research group we have only a few users, three or four, who work closely together. However, most interest is related to end results not system software.

R. Seynaeve How long has the system been in use?

C. Richardson About a year, but with some components, such as graphics, already available.

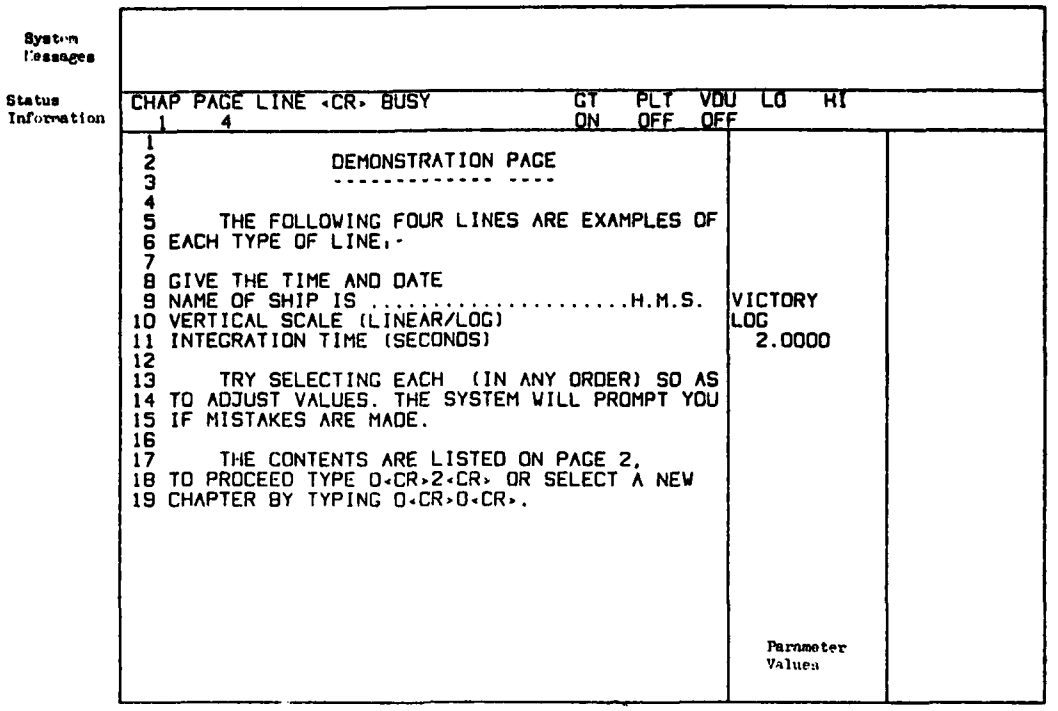


FIG. 1 A TYPICAL PAGE

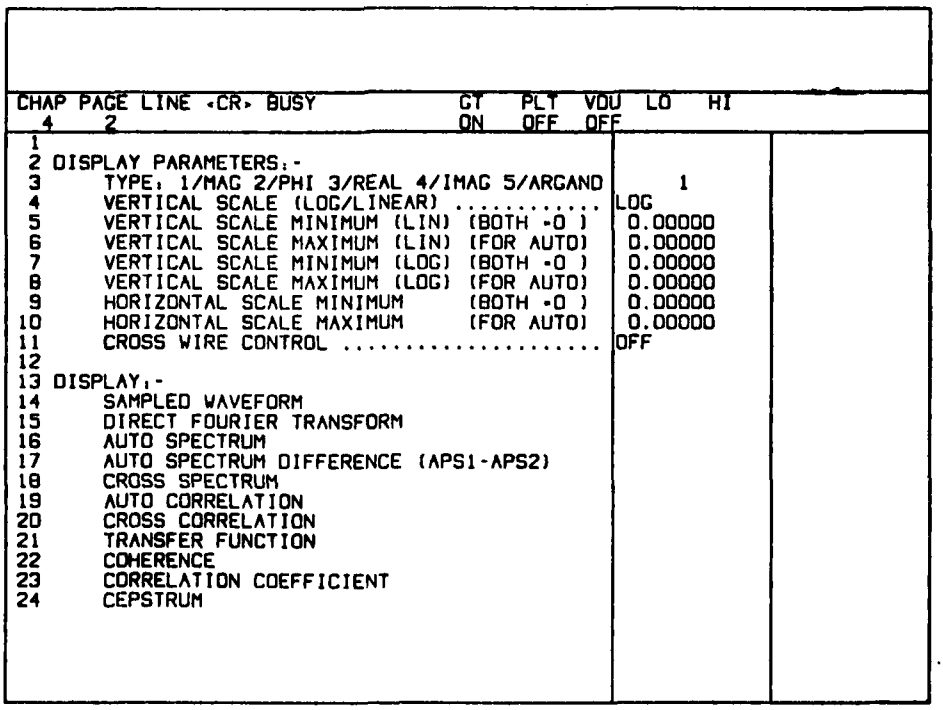


FIG. 2 A PAGE CONTROLLING NARROW BAND DISPLAY PROGRSM

SYSTEM FUNCTIONAL MIGRATION - HIGH LEVEL LANGUAGE TO HARDWARE

Y. S. Wu and G. R. Lloyd
Naval Research Laboratory
Washington, D. C. 20375

Abstract

This paper intends to examine system functional allocation among available processing resources in a systematic way. The discussion will focus first on the criteria for assigning functions to processors, that is, horizontal migration. Then the implementation of "trade-off" choices of implementing a given algorithm in a particular processor in either high level language, assembler language, microprogram, or hardware will be examined (vertical migration). A quantitative measure, based on duty cycle, file size, and program size of the functional algorithm, is derived for vertical migration.

Introduction

Computer science is not a science. A decade to five years ago this author may have been a voice crying in the wilderness. Of late, some big name computer science gurus have begun to make the same assertion. It is at best a pseudo science, embracing metaphysical principles and perhaps scientific methodology to deal with human behavior and resource management in the materialistic world of gadgetry. It is fitting and proper to rename the field 'computer sociology'. In the soft sciences it is fair to invent metaphysical laws (named after the inventor) when a scientific theory cannot be rigorously derived or proven. Some of these well known laws include: Pareto's Law, Parkinson's Law, Murphy's Law, and, of late, Brooks' Law (not to be confused with Brooks' Bill). Although this author is not a computer scientist, he is willing to follow in the footsteps of these celebrities and invent his own metaphysical principle, Wu's Principle, which paraphrases and contradicts the famous Peter Principle. Wu's Principle states:

> In computer science and software, everyone grossly exceeds his or her level of incompetence.

The point of this paper is that the elegant solution to self destructive software and regeneratively complex problems is most often provided by an intuitive, simplistic and common sense approach, rather than uncertain analysis.

Pareto's Advice to Programmers

Many savants recommend the use of high level programming languages to control the cost of producing and maintaining software. The most common objection raised to the use of high level languages is the cry of 100% efficiency. This is very foolish. The nineteenth-century Italian economist Pareto observed that the relationship between utility and volume is 80/20: 80% of the value is derived from 20% of the transactions. In management, 80% of the useful information is derived from 20% of the total paper flow. The advent of the Xerox machine and the electronic computer have simply altered the percentages: the modern ratio is 90/10. In programming terms at least 90% of the software for a system should be coded in a high level language; the remaining 10% might be machine dependent or so heavily used that a faster and more expensive implementation technique might be used.

This observation should have an important influence on the system designer. Solve the problem using all the tools of high level languages before building esoteric hardware or cluttering up a microprogrammed processor with useless (but interesting) special purpose opcodes. At the very least, the designer might learn what the system <u>really</u> must do and where bottlenecks occur.

The first round of mistakes should be as inexpensive and educational as possible. If the rules stated below are followed, different implementations of the same functions can be substituted as required without disturbing parts of the system which appear to work properly. If disaster strikes, it is a software disaster (par for the course) rather than a system disaster which can be very damaging to the professional pride of the designer.

Migration

When a software system designed to run on a single processor exceeds the computing capacity of the processor, two simple courses of action are open. The first is to offload the overworked processor by introducing one or more functionally dedicated processors to handle part of the computing load. The second is to speed up the execution of the critical functions by introducing carefully written assembly language routines, microcode, or additional hardware support. In both cases, the high level language viewpoint can be profitably applied to simplify the programming of the entire system.

If high level languages are good for software cost control they should also be used as a basis for simple system design. When forced to use multiple processors and multiple levels of system implementation techniques,

only two high level language features are used to build the system control structure: procedure call and process to process message passing. The programmer should not care how or where a procedure call is executed or the location of the process receiving or transmitting a message. Let the runtime environment do the schlepp work.

Horizontal Migration

If the critical functions are well defined and have specialized requirements (as in signal processing) the obvious approach is to introduce a dedicated processor which performs a few functions very efficiently. All of the work to be done is defined by the procedure name and the parameter values passed by the call or by the content of the process to process message. As a rule of thumb, communications overhead should be less than 10% of the processing time gain achieved by farming out the process to a subprocessor (G&A for managing a subcontract should not exceed 10% or someone deserves to be fired).

Vertical Migration

Vertical migration chooses an implementation technique within a single processor. Typical levels of implementation are high level language, assembly language, microcode in writable control store (flabbyware), microcode in fixed control store (firmware), and hardware. The rule for determining how different functions are to be implemented is really a rule for calculating return on investment. Simple algorithms which execute in tight loops are the best candidates because minimal investment in specialized implementation can have a high payoff.

W-Test

The following rule has been successfully applied in the design of complicated signal processing systems, where functional routines are well defined arithmetic kernals. It is also applicable where the functions are non-arithmetic but well defined (such as operating system kernals).

Let:

D = functional routine duty cycle (fraction of total load)

A = number of elements in operand arrays (number of data points)

P = number of instructions in functional routine

Then the migration figure of merit W is:

$$W = A*D/P$$

Normalize W=1 when total data and program memories are consumed with 100% execution duty cycle within the processor.

By applying Wu's conjecture of 90/10, the following functional migration rules of thumb are derived:

i) program in high level language when W is less than 0.1
ii) program in assembly language when W is 0.1 to 1 and D less than 0.1
iii) microprogram when W is greater than 1 and less than 10, or when W is 0.1 to 1 and D greater than or equal to 0.1
iv) implement in hardware when W is greater than 10

The W-Test is dangerously simple. Given a hand calculator, even a bureaucrat can find some embarrassing tradeoffs in system design.

Conclusion

This paper is written in simple terms for all levels of management. The key observation is that the optimal solution to an undefined problem is not a complex kludge, but rather no solution at all. A manager may encounter cases where no data is available to apply any of the recommended tests and tradeoffs. Here the course of action is very straightforward: hire a new staff.

For a migration example, see:

A. van Dam, G. Stabler, and R. Harrington, "Intelligent satellites for interactive graphics," Proc. IEEE, vol. 62, no. 4, pp. 483-492, April 1974.

The following pages of visual aid texts were given in support of this paper (Ed.)

SYSTEM FUNCTIONAL MIGRATION

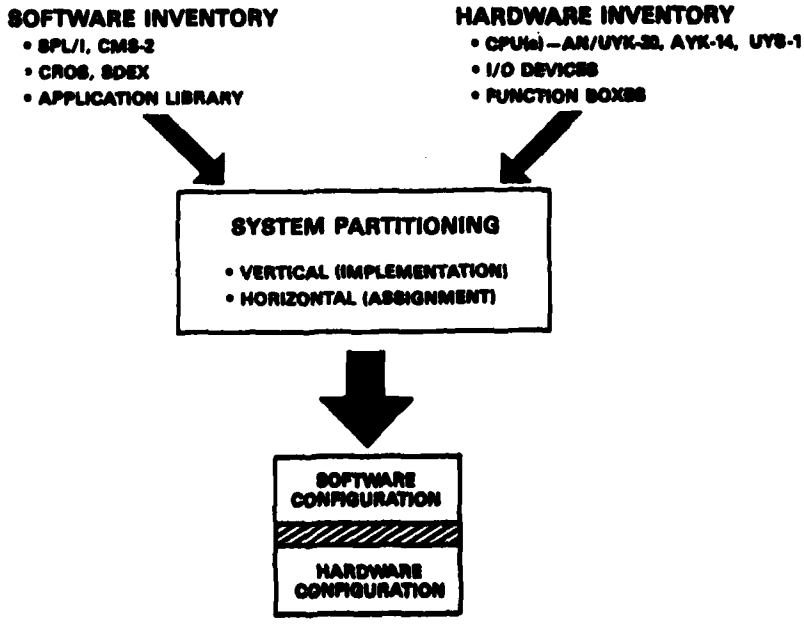
HIGH LEVEL LANGUAGE

TO

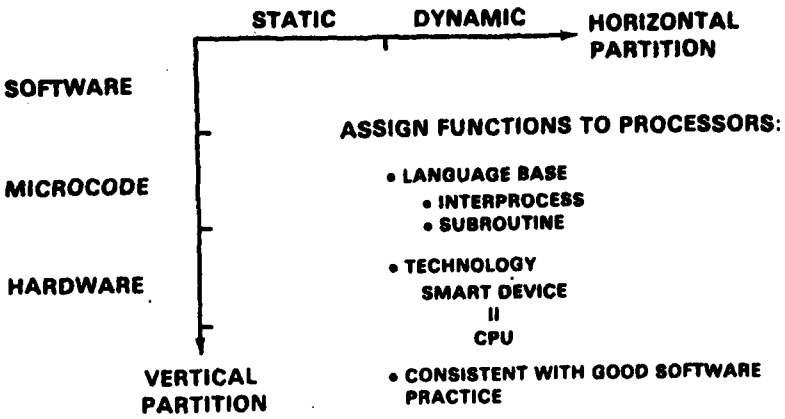
HARDWARE

WU's PRINCIPLE

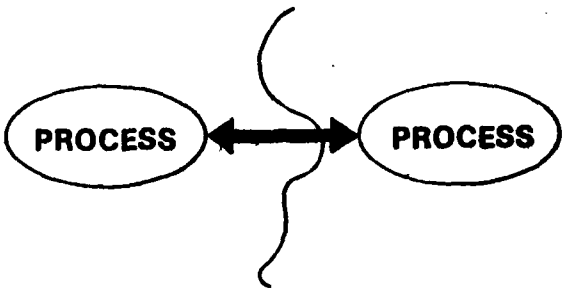
IN COMPUTER SCIENCE AND SOFTWARE, EVERYONE GROSSLY EXCEEDS HIS OR HER LEVEL OF INCOMPETENCE.



HORIZONTAL MIGRATION

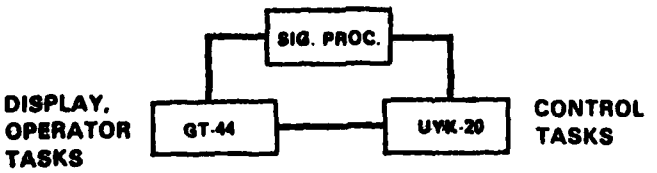


STATIC ASSIGNMENTS

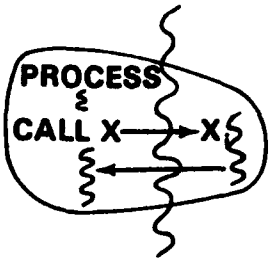


HORIZONTAL PARTITIONING EXAMPLES

PROCESS ⟷ PROCESS (STATIC) SIG. PROCESSING TASKS



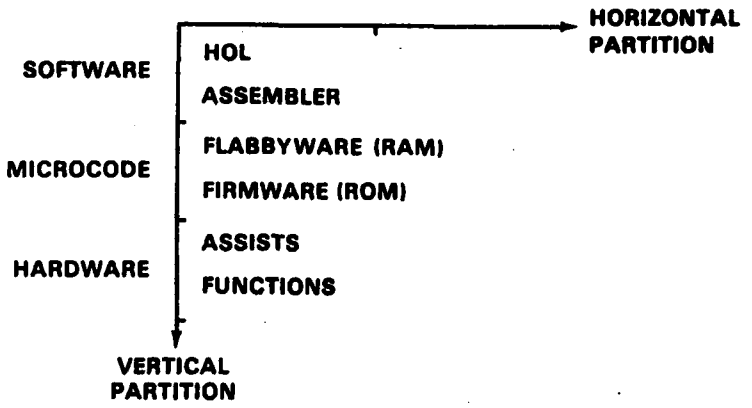
DYNAMIC ASSIGNMENT



EXAMPLE:

SPL/I CONTROL OF DISTRIBUTED
SIGNAL PROCESSING ARCHITECTURE

VERTICAL MIGRATION



CHOOSE AN IMPLEMENTATION APPROACH:

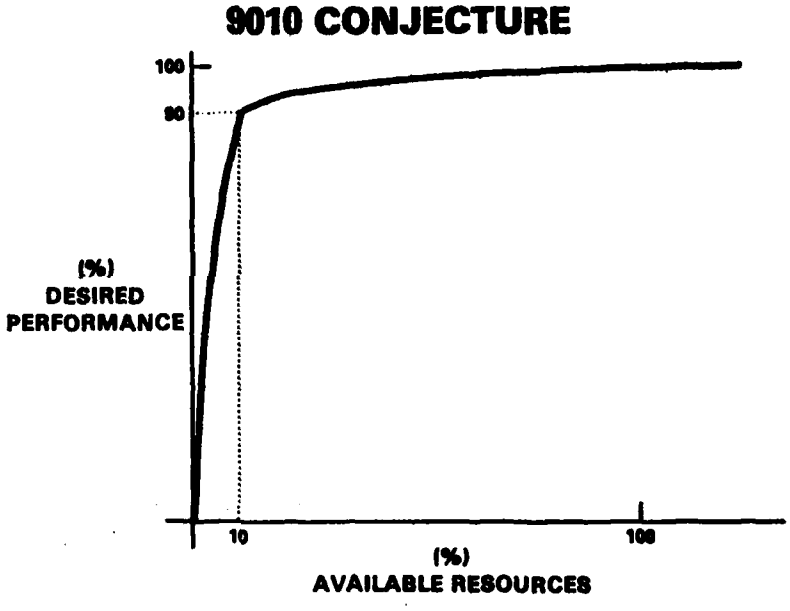
- THROUGHPUT REQUIREMENTS
- DEVELOPMENT COST
- MAINTENANCE COST

JUSTIFICATIONS FOR MICROPROGRAMMING

- HIGH DUTY CYCLE
- SMALL KERNEL
- LONG DATA ARRAYS

WU'S CONJECTURE OF 9010

- EXTENSION TO PARETO'S LAW
- SOCIOLOGY
- ECONOMY
- BUREAUCRACY
- HARDWARE
- SOFTWARE



9010 CONJECTURE FOR SOFTWARE

- 90% OF THE PROGRAMS RUN 10% OF THE TIME
- 10% OF THE PROGRAMS RUN 90% OF THE TIME

ω-TEST

LET:

DC = FUNCTIONAL ROUTINE DUTY CYCLE

|A| = NO. ELEMENTS IN OPERAND ARRAYS

|P| = NO. INSTRUCTIONS IN FUNCTIONAL ROUTINE

THEN:

$$\omega = \frac{|A|}{|P|} \times DC$$

ω NORMALIZATION

$\omega = 1$

WHEN TOTAL DATA AND PROGRAM MEMORIES ARE USED WITH 100% DUTY CYCLE

MICROPROGRAM

WHEN:

(A) $\omega > 1$

(B) $1 \geq \omega \geq 0.1$ AND $DC \geq 0.1$

ASSEMBLER PROGRAM

WHEN $1 \geq \omega \geq 0.1$ AND $DC < 0.1$

HLL PROGRAM

WHEN $0.1 > \omega$

HARDWARE IMPLEMENTATION

WHEN $\omega > 10$

ASP EXAMPLE

FFT:

|A| = 1,024
|P| = 70
DC = 0.2 (20%)

$$\omega = \frac{1,024}{70} \times 0.2 = 2.9$$

RECURSIVE FILTER:

|A| = 512
|P| = 60
DC = 0.4 (40%)

$$\omega = \frac{512}{60} \times 0.4 = 3.4$$

MORE ASP EXAMPLES (Continued)

HARDWARE IMPLEMENTATIONS

"BUTTERFLY"
VECTOR AND FFT ADDRESSING
DeMUX

HORIZONTAL MIGRATION RULE OF THUMB

COMMUNICATIONS OVERHEAD SHOULD BE LESS THAN 10% OF THE PROCESSING TIME GAIN ACHIEVED BY FARMING OUT THE PROCESS TO A SUBPROCESSOR

"It must be remembered that there is nothing more difficult to plan, more doubtful of success, nor more dangerous to manage, than the creation of a new system. For the initiator has the enmity of all who would profit by the preservation of the old institutions and merely lukewarm defenders in those who would gain by the new ones."

Machievelli, "The Prince", (1513)

SIGNAL PROCESSING LANGUAGE AND OPERATING SYSTEM

by

S. Weinstein
Naval Research Laboratory
Washington, D.C., U.S.A.

ABSTRACT Signal Processing Language/One (SPL/I) was developed at the Naval Research Laboratory (NRL) as a high-level computer programming language for signal processing. SPL/I is the first high-level language to be designed for real-time signal processing applications and selected for production use in the U.S. military.

Signal Processing Language/One (SPL/I) is a high level computer language developed primarily for use in signal processing. The language was developed at the Naval Research Laboratory (NRL) and is the first high-level language to be selected for production use in a real-time signal processing application by the United States military. The language is a very flexible block structured language. It contains a wide variety of flow-of-control statements, a rich set of data declarations, and a thorough set of multiprocessing creation and synchronization primitives.

The multiprocessing, or more properly, the multiprogramming features, include process syncrhonization in the runtime environment through the use of counting semaphores and binary semaphores (SPL/I resource variables). SPL/I also provides the user with the ability to terminate and activate processes.

The flow-of-control statements in SPL/I include the definition and invocation of valued and non-valued procedures, the WHILE, UNTIL, and FOR looping statements, the conditional IF-ELSE statement, and the alternative selection CASE statement. The remaining executable statements include an assignment statement, limited GOTO, structured loop termination statements, and an empty statement

The language enables the user to manipulate standard primitive data types (e.g., integers and bit strings). The user can define and use extended data types consisting of groups of like data (ARRAYs) and groups of unlike data (STRUCTUREs) in a simple manner. These arrays and structures can be referenced by accessing single elements, or the entire structure or array. Subsections of arrays can be referenced by a method called array "slicing". The user can declare variables both of dynamic (AUTOMATIC) and static storage classes, and, under complete programmer control, can perform runtime allocation and deallocation of storage for variables. This flexibility allows the user to control and perhaps reduce storage requirements for many problems.

The commenting construct within SPL/I is one of nested square bracket pairs. This, along with the free format structure of the language, makes the flow-of control in SPL/I software easy to understand and encourages self documentation.

To enhance reliability in detecting program errors and again encourage self-documenting code, SPL/I requires explicit definitions of data types and variables, the compatibility of data types in operations, and explicit conversions between data types. These are all checked at compile time, thus eliminating a large class of possible error conditions.

SPL/I requires runtime support for the multiprocessing and dynamic storage features of the language. To prevent the proliferation of different executives for different SPL/I target computers, the Common Real-Time Operating System (CROS) was developed by NRL as the common SPL/I executive. In order to aid in the transportability, CROS is itself written 93% in SPL/I.

CROS offers a variety of process scheduling and dispatching facilities. The scheduling options include process creation on a demand basis, on a periodic basis, and in response to asynchronous events.

CROS is structured such that it is possible to select a subset of its many features that is right for a particular system; thereby eliminating storage and time penalties that would otherwise be incurred by unused language support.

Other features of CROS include high-level I/O and error handling capabilities. For system design evaluation, CROS offers historical data gather on event occurrences and timing information on sections of the application system.

After the user has coded part or all of a system the code is then run through the SPL/I Compiler where semantic and syntax checks are made and object code generated. This object code is then stored in a library by the SPL/I librarian. The user then runs the SPL/I Linkage Editor where the application system is built and a load image built. The user now has the opportunity to specify various attributes of the system and libraries to search to resolve external references. In addition, the Linkage Editor will choose the smallest version of CROS that will handle all the features the user has specified for the system. The user now can either run the system on a simulator or on the machine itself.

Production use of SPL/I with CROS began in 1977 involving two platforms running with the AN/UYS-1 Advanced Signal Processor; additional platforms will be added later. Effort is now complete for retargeting SPL/I and CROS for the Navy standard minicomputers AN/UYK-20 and AN/AYK-14.

The following pages of visual aid texts summarize the language and its application (Ed.)

DISCUSSION

J.E. Vernaglia Does SPL/I or CROS give the programmer any assistance in setting up control blocks for the storage transfer controller and arithmetic processor?

S. Weinstein Yes, and with the Modular Signal Processing Software Development System currently under development, the capabilities will be even further enhanced.

J.E. Vernaglia Did the conversion from assembly language to SPL/I include all P-3 processing modes, or just LOFAR?

S. Weinstein No, but more than just LOFAR.

Y. Lundh You said SPL/I and CROS lend themselves to multi-processing. How does SPL/I and/or CROS know what the multi-processor looks like?

S. Weinstein Currently, CROS does not address the multi-processor configuration.

SPL/I

A HIGH LEVEL LANGUAGE FOR DIGITAL SIGNAL PROCESSING

SPL/I OVERVIEW

BLOCK STRUCTURED

STRONGLY TYPED

SCALAR AND ARRAY OPERATIONS

FACILITIES FOR INDEPENDENT PROCESS EXECUTION AND CONTROL

FEATURES NEEDED FOR DIGITAL SIGNAL PROCESSING, GRAPHICS, AND SYSTEM PROGRAMMING

TARGET COMPUTER INDEPENDENT

DECLARATIONS

VARIABLE	Named memory location of a particular data type
MODE	Programmer-declared data type
PROCEDURE	Block of code invoked as a valued function or non-valued subroutine
PROGRAM	Block of code activated as independent process

DECLARATIONS

EXAMPLES

```
VAR A, B, C: INT;

MODE INT_ARRAY_3:  INT ARRAY(*,*,*);
VAR M:  INT_ARRAY_3 OF SIZE(3,4,5);

PROCEDURE SIN(X:  FLOAT INPUT):  FLOAT;
    [sine computation]
    • • •
ENDPROCEDURE SIN;
```

VARIABLE DECLARATION

NAME	Name by which programmer refers to variable
STORAGE CLASS	Time of allocation of variable
MODE	Data type of variable
INITIALIZATION PHRASE	Optional initialization value upon allocation of variable

STORAGE CLASSES

AUTOMATIC	Storage allocated at block entry and deallocated at block exit
STATIC	Storage allocated before program execution and deallocated at program termination
GLOBAL/EXTERNAL	Static Allocation but accessible to multiple modules
ALLOCATED VARIABLES	Storage allocated dynamically under programmer control

PRIMITIVE MODES

ARITHMETIC

INT	Integer
DINT	Double precision integer
FLOAT	Real
FRAC	Fraction
CINT	Complex integer
CFLOAT	Complex real
CFRAC	Complex fraction

PRIMITIVE MODES

LOGICAL

BOOL	Logical (TRUE, FALSE)

STRING

BIT	Bit string
STRING	Character string

MULTIPROCESSING

PROCESS	Process identification
RESOURCE	Logical or physical resource

MODE DECLARATION

ARRAY	Homogenous data aggregate
STRUCTURE	Non-homogeneous data aggregate
POINTER	Pointers to data elements allocated under programmer control

PROCEDURE AND PROGRAM DECLARATIONS

NAME	Name by which procedure is invoked or program is activated as a process
FORMAL PARAMETER LIST	List of names and modes of formal parameters used by procedure or program
RETURN MODE	Mode of return value on valued procedures only
BODY	Declarations and statements of procedure or program

ASSOCIATIONS OF ACTUAL AND FORMAL PARAMETERS

VALUE	Formal parameter is copy of actual parameter value at time of invocation.
INOUT	Formal parameter refers to storage location of actual parameter.
INPUT	Formal parameter refers to storage location of actual parameter. Cannot be modified.

OPERATORS

ARITHMETIC

**	EXPONENTIATION
+,-	UNARY PLUS, MINUS
*,/	MULTIPLICATION, DIVISION
+,-	ADDITION, SUBTRACTION

STRING

||	CONCATENATION

OPERATORS

RELATIONAL

=,/=,<,>,<=,>=

LOGICAL

NOT	LOGICAL NOT
AND	LOGICAL AND
OR, XOR	LOGICAL OR, EXCLUSIVE OR

VARIABLE REFERENCE

«	ALLOCATED VARIABLE REFERENCE
.	STRUCTURE COMPONENT REFERENCE
( )	ARRAY ELEMENT REFERENCE

EXPLICIT MODE CONVERSIONS

```
VAR A, B : STATIC INT:
VAR C : AUTOMATIC FLOAT:
   .
   .
   .
A : = B + INT(C);
```

ARRAY AND SCALAR ARITHMETIC

```
(SCALAR:)    VAR I, J, K: AUTOMATIC INT;
               •
               •
               •
             I := J + K;

(ARRAY:)     MODE INT_ARRAY_2: INT ARRAY(*,*);
             VAR A, B, C:
                 INT_ARRAY_2 OF SIZE(6,6);
               •
               •
               •
             A := B + C;
```

ARRAY SLICING

```
MODE INT_ARRAY_3:  INT ARRAY(*,*,*);
MODE INT_ARRAY_2:  INT ARRAY(*,*);

VAR AR1:  INT_ARRAY_3 OF SIZE(10, 10, 10);
VAR AR2:  INT_ARRAY_3 OF SIZE(10, 10, 5);
VAR AR3:  INT_ARRAY_2 OF SIZE(10, 5);
    •
    •
    •
AR2 := AR1 (*,*,3 THRU 7);
AR1(*,1, 5 THRU 8) := AR2(*,3, 2 THRU 5);
AR2(9,*,*) := AR3;
    •
    •
    •
```

EXECUTABLE STATEMENTS

DATA MOVEMENT AND COMPUTATIONAL
- *ASSIGNMENT*

PROCEDURE/CONTROL
- *PROCEDURE INVOCATION*
- *RETURN*

BLOCK/CONTROL
- *BEGIN*
- *EXIT*

CONDITIONAL
- *IF-THEN-ELSE*
- *CASE*

EXECUTABLE STATEMENTS

LOOP/CONTROL
- *WHILE*
- *UNTIL*
- *FOR*
- *EXITITERATION*
- *EXITLOOP*

OTHER
- *EMPTY*
- *GO TO*

EXECUTABLE STATEMENTS

PROCESS MANIPULATION

ACTIVATE
TERMINATE

PROCESS SYNCHRONIZATION

REQUEST
RELEASE

CROS

COMMON REAL-TIME OPERATING SYSTEM

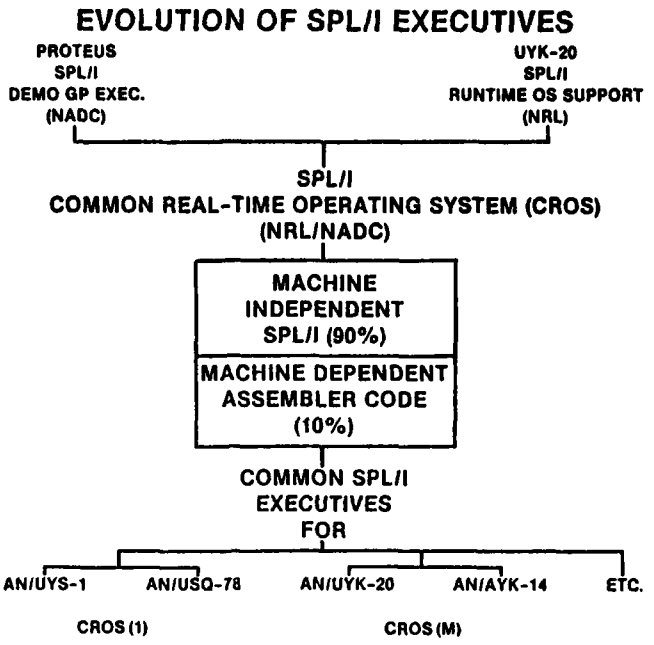
FOR

SPL/I

CROS MOTIVATION

- **NEED FOR EFFICIENT SUPPORT OF SPL/I "MULTIPROCESSING" STATEMENTS AND FACILITIES**

- **POTENTIAL FOR PROLIFERATION OF SPL/I EXECUTIVE ENVIRONMENTS FOR DIFFERENT TARGET COMPUTERS AND TARGET COMPUTER CONFIGURATIONS**



CROS FEATURES

- VARIETY OF SCHEDULING AND DISPATCHING FACILITIES
- ACCESSIBILITY OF FACILITIES FROM BOTH SPL/I AND ASSEMBLY LANGUAGE APPLICATIONS PROGRAMS
- EXTENSIVE SET OF OPTIONS FOR SUPPORT TRADEOFFS AND REQUIREMENTS
- PRE-RUNTIME SPECIFICATION AND GENERATION OF PROCESS ATTRIBUTES FOR RUNTIME EXECUTION SAVINGS
- AUTOMATIC SELECTION OF SMALLEST VERSION WITH REQUIRED FACILITIES

CROS DESIGN GOALS

- OPERATING SYSTEM ENVIRONMENT FOR ALL SPL/I TARGET COMPUTERS—INITIALLY AN/UYS-1, AN/USQ-78, AN/UYK-20, AN/AYK-14
- SUPPORT FOR A WIDE RANGE OF OPERATING SYSTEM FACILITIES FOR TACTICAL SIGNAL PROCESSING AND "REAL-TIME" APPLICATIONS
- TARGET MACHINE-INDEPENDENT USER INTERFACE
- TARGET MACHINE-INDEPENDENT IMPLEMENTATION
- EXTENSIBILITY OF I/O FACILITIES

CROS PROCESSES

- SCHEDULING:
  - ON DEMAND BY ANOTHER PROCESS
  - PERIODICALLY
  - ON EVENT OCCURRENCE
- DISPATCHING:
  - PRIORITY-DRIVEN, 255 LEVELS
  - USER-SELECTED POLICY:
    "NORMAL" - RUN TO COMPLETION OR BLOCKAGE
    "ROUND-ROBIN" - PROCESSOR ALLOCATED EQUALLY
- COORDINATION:
  - VIA SPL/I LANGUAGE FACILITIES

CROS SERVICE ROUTINES

- PROCESSES:
  ACTIVATE, TERMINATE, ENABLE, DISABLE, SUSPEND, UNSUSPEND
- RESOURCES:
  REQUEST, RELEASE
- COUNTING SEMAPHORES:
  INITIALIZE, P, V
- FREE STORAGE:
  ALLOCATE, DEALLOCATE

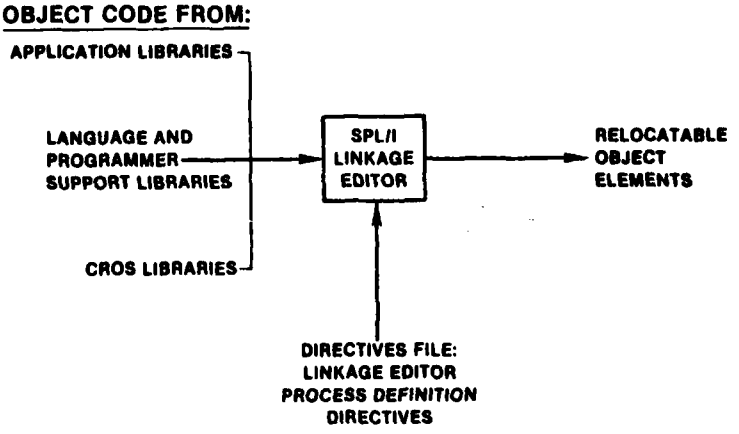
CROS EXECUTIVE SERVICE ROUTINES
(CONTINUED)

- HIGH LEVEL INPUT/OUTPUT
- ERROR HANDLING
- PERFORMANCE MONITORING
- EVENT RECORDING/TIMING

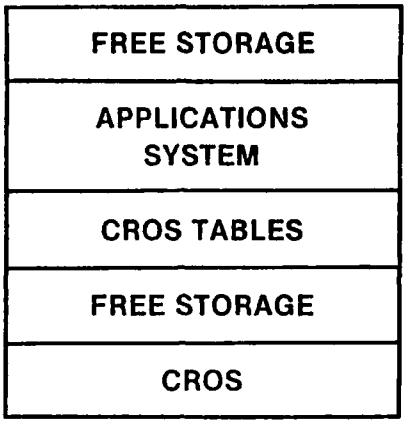
CROS I/O

- FIRM INTERFACE SPECIFICATION FOR I/O DEVICE DRIVERS:
  - REQUEST I/O ROUTINE
  - WAIT ON I/O ROUTINE
  - HALT I/O ROUTINE
  - I/O INTERRUPT ROUTINE
- HIGH LEVEL (SPL/I) USER I/O PRIMITIVES
  - OPEN/CLOSE
  - READ/WRITE
  - CONTROL

CROS LINK EDIT TIME



CROS RUN TIME ORGANIZATION



LINKAGE EDITOR PROCESS DEFINITION DIRECTIVES

- DEFINE CROS FACILITIES REQUIRED BY SYSTEM
- DEFINE NAMES AND CHARACTERISTICS OF SYSTEM
- DEFINE SPECIAL PROCESSING REQUIREMENTS OF SYSTEM

CROS CONFIGURATION OPTIONS

- MULTIPROCESSING/NONE
- PERIODIC PROGRAMS/NONE
- SELECTED EVENT PROGRAMS/NONE
- TERMINATIONS/NONE
- RESOURCE VARIABLES/NONE
- SEMAPHORES/NONE
- SYNCHRONIZATION VARIABLE IMPLEMENTATION OPTIONS
- HISTORY OPTIONS
- ERROR HANDLING OPTIONS

A MODULAR SIGNAL PROCESSING SOFTWARE DEVELOPMENT SYSTEM

by

Peter A. Rigsbee
Naval Research Laboratory
Washington, D.C., U.S.A.

ABSTRACT

A software development system is being built by the U.S. Naval Research Laboratory (NRL) to aid production of modular signal processing software for U.S. Navy airborne ASW platforms using the Navy's Advanced Signal Processor (ASP). This system, part of the ASP Common Operational Software (ACOS) project, is intended to reduce the costs of programming, maintaining, and modifying ASP software. The development system includes the SPL/I support software system, the ACOS Program Generator, and the ACOS Shell. SPL/I is a high-level language designed for real-time signal processing applications. CROS, the standard SPL/I operating system, supports the multiprogramming and data management features of the SPL/I language. SPL/I and CROS are being used by the Navy in signal processing applications. The ACOS Program Generator (APG) is the key component of the development system. It translates software written in a high-level, ASP-independent block language into SPL/I code to be processed by the SPL/I Compiler. This language allows the programmer to speak in terms of bulk storage variables, queues of data buffers, and parameterized signal processing primitives such as "FFT" or "AGC". The APG then takes the responsibility for generating or executing, as appropriate, code to manage storage and to initiate data transfers and complex functions; effectively hiding the underlying hardware architecture from the programmer. The ACOS Shell, which serves as an extension to CROS for the ASP, is the runtime key to the system. The Shell controls and manages access to ASP resources and is the runtime signal processing controller. The Shell is accessed only by software generated by the APG, and the programmer is never aware of the distribution of functions among CROS, the Shell, and the hardware.

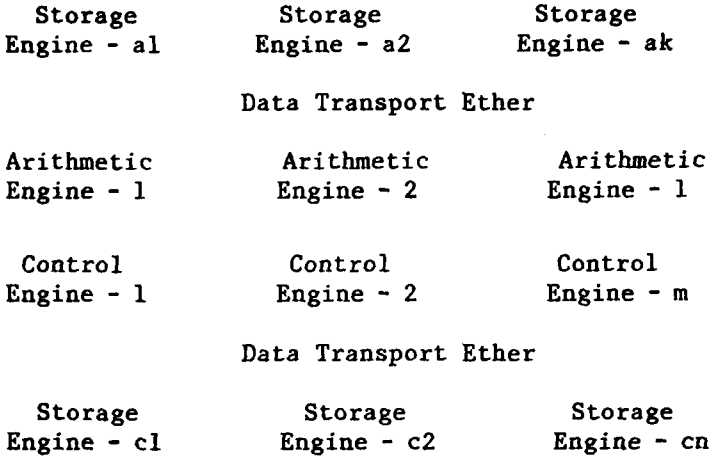
ACOS PROGRAM GENERATOR FACILITIES

The ACOS Program Generator (APG) is a programming tool that allows the implementation of signal processing software in a target machine independent manner. The APG converts its input into SPL/I code that accesses CROS and the ACOS SHELL (see below) for services. The APG will be responsible for:

- Allocation of storage
- Transfer of data between storage areas
- Creation of control information for the SHELL
- Scheduling signal processing and data transfer operations

PROGRAM GENERATOR USER ARCHITECTURE

The ACOS user has the following view of a generic signal processor (GSP) architecture for which he is to write programs:



There is some number of independent Arithmetic Engines, each of which is used to perform signal processing operations. There is some number of Storage Engines which can be accessed by Arithmetic Engines via a Data Transport Ether. These Storage Engines are used to buffer incoming data, partially processed data, and outgoing data as needed. Arithmetic Engines cannot access other Arithmetic Engines. Any Arithmetic Engine-accessible GSP storage not in Storage Engines is considered to be local to an Arithmetic Engine. This local storage (if any) associated with an Arithmetic Engine is allocated by the ACOS system (APG and/or SHELL).

There is some number of independent Control Engines, each of which is used to perform control operations. There is some number of Storage Engines which can be accessed by Control Engines via a Data Transport Ether. Storage Engines are used to buffer incoming data, partially processed data, and outgoing data as needed by Control Engines to determine control actions. Control Engines effect control through a Control Ether (not shown in the diagram) that connects to all GSP engines. Control Engines cannot access other Control Engines except through the Control Ether. Any Control Engine-accessible GSP storage not in Storage Engines is considered to be local to a Control Engine. The local storage (if any) associated with a Control Engine is allocated by the ACOS system (APG and/or SHELL).

An ACOS user first writes a number of ENGINE PROGRAMs. ENGINE PROGRAMs are the basic units of software that are executed by the GSP. They specify control processing or signal processing to be performed by the GSP to meet operational requirements. The user specifies how ENGINE PROGRAMs are related to one another by defining a set of GRAPHs. These GRAPHs specify

the topological connections among ENGINE PROGRAMs (ENGINE PROGRAMs can be viewed as the nodes of GRAPHs) by identifying queues to be used to exchange data between pairs of ENGINE PROGRAM executions or between an ENGINE PROGRAM execution and the outside world. This user software, comprised of ENGINE PROGRAMs and GRAPHs, is processed by the APG and the SPL/I Compiler and loaded into the target hardware.

At runtime the GSP executes ENGINE PROGRAMs under the supervision of the SHELL. Each instantiation of an ENGINE PROGRAM is executed as an independent task by exactly one GSP engine. A GSP engine executes at most one task at a time. When a GSP engine completes execution of a task, the task and its associated local storage is destroyed. Two engines can concurrently execute distinct tasks that are instantiations of the same ENGINE PROGRAM. Tasks are created by the SHELL in response to the presence of data in the queues specified in GRAPHs. An instance of a GRAPH (called a GRAPH task-set) is created at runtime by STARTing the GRAPH. STARTing the GRAPH causes the SHELL to begin monitoring the GRAPH-specified queues. Upon discovering the presence of sufficient data in some of the queues, the SHELL will create tasks to execute GRAPH-specified ENGINE PROGRAMs.

Tasks are differentiated between control tasks and arithmetic tasks. Control tasks are executed by Control Engines and arithmetic tasks are executed by Arithmetic Engines. Only control tasks can start and stop GRAPHs. Consequently, ENGINE PROGRAMs are called control ENGINE PROGRAMs for arithmetic ENGINE PROGRAMs depending upon the type of GSP engine on which they are to execute. Similarly, GRAPHs are either control GRAPHs or arithmetic GRAPHs. Initially at runtime a user-specified control GRAPH is STARTed by the ACOS system. This initiation causes the SHELL to create a control task-set. This task-set may then proceed to start other control GRAPHs or arithmetic GRAPHs defined by the ACOS user.

The numbers of Arithmetic Engines, Control Engines, and Storage Engines may differ among specific implementations of the GSP. The Storage Engines that a particular Arithmetic Engine or Control Engine can access may also differ among specific implementations of the GSP. The engines are not necessarily homogeneous. Associated with each GSP engine is a set of capabilities that the engine can support. For example, an Arithmetic Engine may be capable of executing only certain arithmetic primitives. A Storage Engine may be capable of storing only variables of particular SPL/I modes. The particular capabilities associated with an engine depend upon the specific implementation of the GSP.

PROGRAM GENERATOR USER FACILITIES

The definitions of the APG facilities are given below.

Engine Identifiers:

Arithmetic Engine, Control Engine, and Storage Engine identifiers are of two types: engine designators and engine class designators. An

engine designator references a particular Arithmetic Engine, Control Engine, or a Storage Engine. An engine class designator references a specific set of Arithmetic Engines, Control Engines, or Storage Engines. Engine identifiers are used by the APG and the SHELL to assign execution and storage resources. In the event an engine class designator is used, then the ACOS system will select one engine from the specified set.

Storage Template Declarations:

FIELD

Defines size of field used in one or more block definitions. Size defined in bits.

BLOCK

Defines data template. FIELD and BLOCK pertain only to the logical layout of data. Entities declared using FIELD and BLOCK cannot be accessed using ordinary SPL/I features; thus SPL/I packing is neither assumed nor enforced. BLOCK templates are used to define the basic data entities that comprise instances of QUEUEs and of BUFFERs.

QUEUE STRUCTURE

Defines queue data template. QUEUE STRUCTURE names are used to define QUEUE variables and formal inputs and formal outputs of ENGINE PROGRAMs and GRAPHs. QUEUE STRUCTUREs represent FIFO queues of data items defined using a common BLOCK template.

Variable Declarations:

BUFFER

Defines the names for BUFFER variables. BUFFER variables exist only in Arithmetic Engine and Control Engine local storage; BUFFER variables are used for data exchange among PRIMITIVEs. A BUFFER variable has two components: control information and a data element (an instance of an underlying BLOCK template). Storage for control information may be allocated (when an ENGINE PROGRAM task is executed) and freed (when the ENGINE PROGRAM task completes) as a result of the BUFFER declaration. Data element storage is allocated and freed by execution of BUFFER operations listed below.

Operations on BUFFER variables:

CREATE – create storage for the data element of the specified BUFFER variable

REPLACE – write a PRIMITIVE output into the data element of the specified (and previously CREATED or INSERTed) BUFFER variable

INSERT – create storage for the data element of the specified BUFFER variable and write a PRIMITIVE output into it

REMOVE – read the contents of the data element of the specified BUFFER variable as a PRIMITIVE input and destroy the data element storage upon completion of the read

COPY – read the contents of the data element of the specified BUFFER variable as a PRIMITIVE input

DESTROY – destroy previously CREATED or INSERTed data element storage for the specified BUFFER variable

QUEUE

Defines the names for QUEUE variables. QUEUE variables can exist only in Storage Engines. QUEUE variables are used for data exchange among ENGINE PROGRAM tasks. A QUEUE variable has two components: control information and queue elements (instances of the underlying BLOCK template). Storage for control information is allocated (when a GRAPH is STARTed) as a result of the QUEUE declaration. Queue elements are allocated and freed by execution of QUEUE operations listed below.

Operations on QUEUE variables:

ENQUEUE – add specified number of queue elements to the tail of the specified QUEUE variable

DEQUEUE – remove and read specified number of queue elements that are at the head of the specified QUEUE variable

READQUEUE – read specified number of queue elements that are at the head of the specified QUEUE variable

Topological Declarations:

PRIMITIVE

Defines a low level primitive available to an ENGINE PROGRAM. A PRIMITIVE may consist of any number and combination of executions of lower level operations on an Arithmetic Engine or Control Engine. PRIMITIVE declarations are always provided a priori to the APG user.

ENGINE PROGRAM

Defines a sequence of SPL+I code and PRIMITIVE invocations that perform a particular processing algorithm. An instantiation of an ENGINE PROGRAM is called a (control or arithmetic) task and executes on a specified Control Engine or Arithmetic Engine. Any number of instantiations of an ENGINE PROGRAM can exist concurrently. QUEUE

variables are the inputs and outputs of tasks. A task is scheduled by the SHELL when the number of queue elements on each of its input QUEUE variables meets or exceeds the threshold specified for that QUEUE variable.

NODE

Associates actual QUEUE variables with formal QUEUE parameter names for a specified ENGINE PROGRAM.

GRAPH

References a set of ENGINE PROGRAMs and declares the QUEUE variables that serve as the inputs and outputs of ENGINE PROGRAM tasks. A GRAPH is a static description of how instantiations of ENGINE PROGRAMs are to exchange data. STARTing a GRAPH creates an instance of the GRAPH (called a GRAPH task-set) and causes the creation of control storage for the QUEUE variables declared within the GRAPH declaration and for the QUEUE variables declared within the ENGINE PROGRAMs referenced within the GRAPH declaration. There is no executable SPL/I code within a GRAPH; it is simply a collection of storage template, QUEUE, and NODE declarations. A GRAPH must reference only arithmetic ENGINE PROGRAMs or only control ENGINE PROGRAMs.

Primitive Statement:

INVOKE

When executed, specifies that execution of a PRIMITIVE should be initiated. The INVOKE includes specification of all the input and output variables required for initiation of a general purpose PRIMITIVE to meet a particular requirement. The QUEUE and BUFFER operations described above are used to access QUEUE and BUFFER variables. The Control Engine or Arithmetic Engine specified in the ENGINE PROGRAM declaration containing the INVOKE will execute the INVOKEd PRIMITIVE.

Control Statements:

START

When executed, specifies that an instance of a GRAPH (called a task-set) is eligible for scheduling. An already STARTed GRAPH task-set cannot be reSTARTed without an intervening STOP of the GRAPH task-set.

STOP

When executed, specifes that a GRAPH task-set is no longer eligible for scheduling. In the event a control task-set is STOPped, then all GRAPH task-sets STARTed by that control task-set are also STOPped.

Execution:

Runtime entities:

An instantiation of an ENGINE PROGRAM is called a (control or arithmetic) task and executes on a specified Control Engine or Arithmetic Engine. QUEUE variables are the inputs and outputs of tasks.

Task Queue Discipline:

A task is scheduled by the SHELL when the number of queue elements on each input QUEUE variable meets or exceeds the threshold specified for that QUEUE variable in the ENGINE PROGRAM's formal parameter list and in the referencing NODE declaration. By its completion a task must have DEQUEUEd exactly the number of queue elements specified for each input QUEUE variable as the consume amount and the task must have ENQUEUEd exactly the number of queue elements specified for each output QUEUE variable as the produce amount.

Computational Determinancy:

Any number of instantiations of an ENGINE PROGRAM can exist concurrently. In the event that an ENGINE PROGRAM's engine identifier is a class designator, then the SHELL preserves the queue data ordering implicit in each GRAPH for concurrent instantiations of the same task-set NODE. (It is sufficient to prohibit multiple concurrent tasks executing the same task-set NODE. It may be useful, however, to allow multiple concurrent tasks executing the same NODE in a task-set. If they do, then "the NODE" must be run as a pipe.)

ACOS SHELL FACILITIES

The ACOS SHELL is the runtime support software of the ACOS system. Development of the SHELL facilities must, of necessity, lag behind that of the APG facilities. The following is a brief description of the main features of the ACOS SHELL.

Monitors

The SHELL consists of a set of runtime engine monitors. Each engine (control, arithmetic, or storage) is represented by a single monitor. Monitors consist of SPL/I processes and procedures that have three functions:

- task scheduling: the initiation of the execution of an ACOS task.
- meassage transmittal: the format and transmittal of a message to some other monitor.
- message receipt: the translation and processing of a message sent by some other monitor.

Lists

Monitors manage two logical lists of resources. One list is that of GRAPHs that can execute on the engine. This list contains information about all the tasks that comprise the GRAPH. The other list is that of QUEUEs stored in the engine. Not all monitors necessarily manage both list types, nor do the number of elements in the lists necessarily bear any relationship between monitors.

Messages

Monitors communicate through the use of messages. The following is a tentative list of SHELL messages.

Message Processing Complete – sent after processing of some other message; specifies that processing has completed and that another message may be sent to this monitor.

Start Graph – specifies that a particular graph may be scheduled. May cause "Connect Queue" messages to be issued.

Stop Graph – specifies that a particular graph may no longer be scheduled, may cause "Disconnect Queue" messages to be issued.

Connect Queue – associates actual queues with a node, graph, or system feature as a sink or as a source.

Disconnect Queue – disassociates actual queues from a particular sink or source.

Add Data – transfers data to a queue. May cause "Data Ready" message to be issued to the queue sink.

Read Data – asks for data to be sent from a queue.

Data Transmit – sends data from a queue in response to a "Read Data" message.

Data Ready – indicates that a queue has sufficient data for task initiation.

Error Processing

Monitors must monitor and process runtime error conditions. The set of error conditions and the processing mechanism is to be determined.

SUMMARY

The APG and ACOS SHELL are intended to raise the level of the programming signal processing software beyond that of standard high level languages. This will both reduce programming cost and, because of the improved understandibility of the resultant operational software, reduce life-cycle maintenance costs. Although designed for the ACOS project, it is of sufficient generality for use in other signal processing application. In addition, because architecture details are buried in the PRIMITIVE definitions and the SHELL implementation, the resulting operational software is highly machine transportable.

DISCUSSION

J.E. Vernaglia Will SPL/I programming still be necessary under the ACOS system?

P.A. Rigsbee Yes, SPL/I code is integrated with APG facilities in the ACOS system. Some functions will need none of the ACOS features and, therefore, will be coded purely in SPL/I.

R. Seynaeve What was the overall cost of SPL/I development?

P.A. Rigsbee Two and half million dollars and five years.

LABORATORY UTILIZATION OF A STANDARD SIGNAL PROCESSOR

by

Frank W. Molino
John E. Vernaglia
Jerry C. Lamb
William J. Coggins
U.S. Naval Underwater Systems Center
New London, Connecticut

ABSTRACT The AN/UYS-1 Advanced Signal Processor (ASP), or PROTEUS, is the U.S. Navy's standard signal processor and is being utilized in several systems. For shipboard use, the ASP is used as a processor with no peripherals, generally passing outputs to an AN/UYK-20 or other general purpose computer for display processing. In a laboratory or advanced development situation, ASP utilization is more difficult since it is not designed for this environment. There is no compiler in the ASP, no interactive peripherals, and investigation of machine status is via toggle switches and light readouts on the maintenance panel.

In order to provide a means of utilizing the ASP effectively in advanced development situations, NUSC developed an ASP laboratory test bed. In this configuration the ASP is supported by a general purpose computer, specifically a DEC PDP-11/45, whose normal features are available to users including editors, utilities programs, etc. In addition, the 11/45 is physically connected to the ASP through two specially designed and constructed interfaces. One connects the PDP's high-speed UNIBUS to a PROTEUS Digital Channel (PDC) using NTDS protocol. The second replaces the maintenance panel's toggle switches and lights by connecting the maintenance panel interface directly to the PDP-11. The heart of the test bed's effectiveness is in the special software written to allow research use of the ASP.

From the user's viewpoint, the entire development process can be controlled from a single PDP-11 terminal using standard computer procedures. This replaces steps of generating tapes for compilation, using a special I/O adapter to load programs, requiring a special data interface, and utilizing switches and lights to debug operational programs. Program development time is reduced dramatically over the previous procedure, and this allows easy generation and testing of advanced signal processing algorithms.

This paper describes the ASP Test Bed and its features, special microcode programs, and the use of the ASP in a specific development situation.

INTRODUCTION

The NUSC, New London, ASP Test Bed was designed to provide a facility for both hosting the Advanced Signal Processor (ASP) and for communicating with

the Facility for Automated Software Production (FASP), located at the Naval Air Development Center (NADC) in Warminster, Pennsylvania. The FASP is the single source for all support software required by the ASP. It allows users to generate and maintain data bases for their individual applications.

The principal components of the ASP Test Bed are a Digital Equipment Corporation (DEC) PDP-11/45 minicomputer with peripherals, hardware interfaces to the ASP, and associated system software. The test bed evolved in three phases, providing a system with increasing sophistication in terms of user and intercomputer interfaces. Phase 1 allowed immediate usage of the ASP hardware upon its arrival at NUSC. The capabilities provided were: creation of source files on the PDP-11 and transmission of them to the FASP; receiving load modules created at the FASP and inputting them into the ASP using the IBM System I/O Adapter (SIOA) (a custom lab tool which allowed input/output operations, via magnetic tape); debugging at the assembly language level was performed via the SIOA and at the microcode level via the maintenance panel (both were crude and time-consuming tasks). Phase 2 automated all maintenance panel functions for the user; e.g., loading and dumping of memories, altering and displaying registers, setting of breakpoints, etc. Control was provided by an interactive command language which the user accesses through a CRT terminal. Phase 3 provided a high-speed I/O interface between the ASP's PROTEUS Digital Channel (PDC) and the test bed's host PDP-11/45 computer to permit ASP utilization of the test bed peripherals. Together with a comprehensive support software package, tailored to this channel, the user now commanded all software development functions, from initial source code typing to operational testing and graphic display, while seated at a single CRT terminal.

A block diagram of the test bed equipment, showing these interfaces, is given in Figure 1. The following sections contain greater detail on the software and hardware developments required to obtain these capabilities. Starting with a brief description of the AN/UYS-1 and concluding with a look at how the test bed is being utilized today by the Light Airborne Multipurpose System (LAMPS) program at NUSC, New London.

1. AN/UYS-1, ADVANCED SIGNAL PROCESSOR

The Advanced Signal Processor (ASP) is a militarized signal processor developed by the Federal Systems Division of IBM Corporation under contract to the Naval Air Development Center, Warminster, Pennsylvania. It has been designated the U.S. Navy standard signal processor. Figure 2 is a high level block diagram of the ASP. A brief description of some of the ASP's subunits follow [1].

1.1 ASP SUBUNIT CHARACTERISTICS

1.1.1 CONTROL PROCESSOR (CP)

The CP is a microprogrammed 16/32-bit general computer which executes system supervisory and data management functions. It contains a 32-bit ALU and operates on a 300-nanosecond microinstruction cycle time. It has 64 general

registers and 192 external registers. Its storage capability consists of 64K x 34 bits of program store and 4K x 34 bits of microprogram control store.

1.1.2 ARITHMETIC PROCESSOR (AP)

The AP is a microprogrammed 32-bit special purpose processor. It contains one or two modular, pipeline Arithmetic Elements (AE) executing microinstructions at a 10 MHz rate, in lock-step. Each AE includes a 16 x 16-bit multiplier and 32-bit, 3-way adder as well as a 2K x 34-bit working store. The AP contains a 2K x 68-bit microstore for arithmetic microprograms and a sine-cosine generator. Each AE is capable of executing 10 million multiplies and 20 million adds per second, completing a 1K complex FFT in 2 milliseconds.

1.1.3 BULK STORE

The Bulk Store is a data and coefficient/parameter memory with a capacity of up to two million 32-bit words. Memory is organized as 64-bit double words with eight bits of error correcting/detecting codes. The basic cycle time is 800 ns, with a 400 ns interleaved cycle.

1.1.4 INPUT/OUTPUT CHANNELS

A mix of up to eight PROTEUS Digital Channels (PDC) or Navy Tactical Data Standard (NTDS) channels are available for high-speed data transfer. The maximum aggragate transfer rate (for all channels) to Bulk Store is 2.5 MHz.

1.2 PROGRAMMABILITY

1.2.1 MICRO PROGRAM

Both the CP and AP are controlled by micro-coded programs. The CP microprogram is not intended to be modified by the user. The AP signal processing micro programs or "macros" are maintained in a library as part of the FASP. New AP macros can be coded and translated by a user through use of support software available in the FASP.

1.2.2 SIGNAL PROCESSING LANGUAGE (SPL)

Application programs are coded in SPL, which is an assembly level language. SPL instructions are emulated by the CP micro program.

1.2.3 HIGH LEVEL LANGUAGE

SPL/1, the U.S. Navy's standard signal processing language, may be used for writing high-level language programs for the ASP. The SPL/1 compiler is accessible via the FASP.

2. ASP TEST BED DEVELOPMENT

Several pieces of hardware and software were designed fabricated/coded, tested, and integrated, to finally arrive at the full scale test bed in use today. Each of these items will be briefly discussed with special emphasis on what capabilities were acquired at each stage in the development cycle.

2.1 COMMUNICATION WITH NADC'S FASP

An initial requirement for the test bed was an ability to access and interact with the Program Generation and Development software maintained solely at NADC. Since all compilers, assemblers, translaters, and linkers needed to generate load modules for the ASP are contained in the FASP, this communication link was imperative. To achieve this capability, only one custom piece of software was required. This program provided for interactive communication necessary for transferring files to and from the FASP. In addition, a previously written software package to allow the PDP-11 to simulate a CDC-200 UT terminal was obtained externally. While no custom hardware was required, a 4800-baud modem and its associated interface to the PDP-11 were procured.

This feature allowed the test bed to function, albeit disjointly, through physical transport of magnetic tapes from the PDP-11 system to the SIOA connected to the PROTEUS Digital Channels of the ASP. Debug operation was very cumbersome and slow, using the SIOA for SPL-level code and the manual maintenance panel for AP microcode and hardware troubleshooting.

2.2 MAINTENANCE PANEL (MP) INTERFACE

The manual maintenance panel of the ASP, pictured in Figure 3, illustrates the type of functions one is able to perform. In an effort to expedite and facilitate the access of both micro-level and SPL-level information available to the user, it was found desirable to control these accesses via software. To this end, an interface was devised to allow the PDP-11 host to perform all possible maintenance panel functions [2], [3].

The interface hardware can be viewed conceptually as capturing the lights and switches of the MP in the form of "read only" and "write only" registers, respectively, for the PDP-11 to access. Key control signals generated by the ASP are also captured and used to generate interrupts for the PDP-11 to service.

The NUSC/NL ASP Test Bed utilizes this hardware in conjunction with a software driver interfacing to a high level command language, both running on the PDP-11 processor. A user of the test bed will therefore be able to easily perform any of the MP functions, via a CRT terminal, in a rapid, straightforward manner.

2.2.1 HARDWARE DESIGN OVERVIEW

The maintenance panel interface logic consists of three major parts: PDP-11 resident logic, IBM logic page extracted from the maintenance panel, and the NUSC designed interface board (MPILB).

One system unit of the PDP-11 is populated with seven DEC modules to effect the interface to the UNIBUS. The DEC modules provide interrupt control, address selection, read/write control, as well as UNIBUS drivers and receivers. The interface logic board (MPILB) communicates directly with this module set.

The MP1 page (IBM 4π technology) [4] was removed from the ASP maintenance panel and mounted within the interface enclosure. The board performs the identical tasks as when mounted in the maintenance panel except, instead of lighting lights and reading switch settings, it is actually communicating with PDP-11 registers. This is performed in a manner which is completely transparent to both the MP1 page and, of course, the ASP itself.

The MPILB is the heart of the PDP-11 to maintenance panel port interface. It provides all necessary logic functions to mate the MP1 page to the UNIBUS interface modules. This interface can be viewed functionally (say, from a PDP-11 system programmer's point of view) by defining the registers which comprise it. Figure 4 illustrates this.

2.2.2 SOFTWARE DRIVER

The purpose of the maintenance panel interface driver is to provide the following services to user programs that access the ASP over the maintenance panel interface.

a. Emulates a block transfer device. This is a standard function for a driver of an interrupt driven, single word transfer device, such as the maintenance panel interface.

b. Allows user programs to access the interface through the use of natural I/O functions. There are numerous encodings and protocols which must be used in performing the various maintenance panel operations. The user program need not be concerned with these, but merely issues commands such as read bulk store, write program store, etc.

c. Presents all I/O functions in terms of operating system calls. This driver is fully integrated into the RSX-11M operating system of the PDP-11/45.

2.2.3 TEST BED COMMAND LANGUAGE

In order to further facilitate access to the functions provided by this interface package, a high-level interactive command language was written [5]. The Test Bed Command Language (TCL) was designed to provide the type of functions itemized below.

a. Examine and change any location in any of the ASP memories.

b. Load files at a specified address.

c. Set and run to breakpoint (both CP and AP).

d. Execute instruction step or microstep.

e. Stop/reset.

f. Execute command files.

These functions are executed by entering simple one or two-character commands at a user terminal. For example, "/" will display a current memory location, "%" will reset the ASP, "#" will cause execution of a command file, etc. Stores and registers are referenced by simple mnemonics. For example, PS - program store, BS - bulk store, CSAR - control store address register, etc. Through use of TCL, debug capabilities of both ASP software/microcode and hardware were greatly enhanced.

2.3 PROTEUS DIGITAL CHANNEL INTERFACE

The NUSC Advanced Signal Processor (ASP) Test Bed includes an interface from a PROTEUS Digital Channel (PDC) to the PDP-11/45, in order that the PDP-11 may provide the following services:

a. Perform a variety of pre and post-processing tasks to run in conjunction with PROTEUS software - for example, tasks which supply synthetic data to the ASP or display its output.

b. Duplicate the SIO Adapter's function as the peripheral from which the ASP is bootstrapped/loaded and from which the diagnostics are run.

c. Function as System Control Unit (SCU) for the purpose of communicating with the ASP according to standard software protocols [6].

2.3.1 PDC INTERFACE CONFIGURATION

As diagrammed in Figure 5, the PDP-11 interface to the serial PDC was implemented with an intermediate conversion to NTDS protocol. This method was chosen since an NTDS type interface for the PDP-11 family of minicomputers would (and did) have use for other projects in the Navy community. In addition, this made the design somewhat simpler than interfacing directly to the PDC. The PDC/NTDS adapter manufactured by General Electric (GE), Syracuse, New York, was available at that time.

The PDP-11 contains a standard DEC DR-11B DMA interface to which a custom set of logic boards (also resident in the PDP-11) communicate and in turn generate the standard NTDS, intercomputer input and output subchannels [7]. This custom logic is cabled to the GE PDC/NTDS adapter box which performs packing (16 to 32 and vice versa), buffering, and protocol conversion to the PDC serial channel. The PDC has sink (input) and source (output) subchannels which are completely independent in operation. However, from the PDP-11 point-of-view operation is basically half-duplex, but no special considerations (i.e., synchronization) need be made since the PDP-11 will queue any concurrent I/O requests and handle them in a sequential manner. This channel configuration was utilized together with several software modules to provide desired services as stated in the preceding section.

2.3.2 SUPPORT SOFTWARE

At the bottom of the software hierarchy which supports channel services is an RSX-11M compatible driver for the PDC interface. It directly supports function (a) of section 2.3 by providing QIO calls which are symmetric to the Execute I/O (XIO) instruction of the ASP. At the next higher software level, two FORTRAN programs have been written for the purpose of remotely bootstrapping PROTEUS from a PDP-11 disk file, and emulating the SCU capabilities of the SIO Adapter. At the next level an RSX-11M command file has been written, which activates these tasks to allow the PROTEUS diagnostics to be run remotely from the PDP-11 [8].

2.3.2.1 REMOTE IMPL/IPL

Initialization of the ASP involves loading micro programs into CP control store and the system loader into program store over a PDC. This process may be activated locally be depressing the IMPL (initial micro program load) button on the ASP or remotely by sending it certain PDC command sequences. Heretofore, the initial load data was contained on mag tape and entered via the SIOA through a series of manual operations. The PDC/NTDS/UNIBUS interface allows the PDP-11 to be used as the load device for the ASP, with the bootstrap information (IMPL/IPL data) stored on a disk file. This procedure has been automated for the user by a straightforward FORTRAN program.

2.3.2.2 SIOA EMULATION

The System Input/Output Adapter (SIOA) is the principal manufacturer supplied peripheral for use with the ASP in a laboratory environment. The SIOA consists of two PDC interfaces, one of which is connected to a magnetic tape unit and line printer, and the other connected to a CRT terminal. It has a maintenance panel with rotary switches, push buttons, and lights to be used in conjunction with these interfaces to exercise control on these devices or initiate data transfers. The SIOA emulation program uses the PDC/NTDS/UNIBUS interface to perform two major SIOA functions. First, provide input and output communication (in hexadecimal format) between any PDP-11 terminal and the ASP without any need for maintenance panel intervention and, secondly, provide magnetic tape control and transfer functions with the capability of simulating the tape with a disk file.

2.3.2.3 REMOTE DIAGNOSTICS

As an application of the support software for the PDC/NTDS/UNIBUS interface, a procedure for running the ASP hardware diagnostics remotely from the PDP-11 has been developed. An RSX-11M indirect command file is used to interactively implement the remote diagnostic procedure. A simulated load tape (disk file) was generated using the manufacturer's supplied diagnostic tape. From a user terminal, the diagnostic routines can be loaded and run with one of three options specified.

a. full fault detection and localization

b. full fault detection without localization

c. run to a specified test and loop on that test

At the completion of the diagnostics, the user will receive a GO/NO GO message along with the suspected fault group call-out if an error was detected.

2.3.3 CHANNEL PERFORMANCE

As a by-product of a demonstration scenario designed to show how the test bed could be used in a real-time signal processing environment, a measurement of the maximum channel (PDC/NTDS/UNIBUS) bandwidth was made. Buffers of length 512 and 1024 words (16-bit) were moved from the PDP-11's memory to the ASP. Runs were made for various durations with the PDP-11 keeping count of the number of buffers transferred. These runs yielded buffer transmit times of 1/96 second and 1/75 second for 512 word and 1024 word buffers, respectively. Since the overhead in performing I/O is independent of buffer size, the difference in transfer times is due entirely to the extra 512 words. Thus, the channel bandwidth is 512/(1/75 - 1/96) = 170,000 words/sec. This rate compares very favorably with the advertised maximum bandwidths achievable with both NTDS and PDC type channels.

3. USE OF THE ASP TEST BED FOR LAMPS SOFTWARE TESTING

The AN/SQQ-28, Light Airborne Multipurpose System (LAMPS) (shipboard system) program at NUSC, New London, is currently utilizing the ASP Test Bed for testing and verification of incremental software deliveries. As will be described in the following paragraphs, this testing makes extensive use of the interfaces previously discussed, as well as the communication link to the FASP. In addition to load modules of LAMPS software being transmitted from the FASP, weekly listings of all Program Trouble Reports (PTR's) are also received.

The configuration for LAMPS software testing employs the PDP-11 as System Control Unit (SCU) and load device via the NTDS/PDC hardware/software package. The load tape to be tested is produced via the FASP and maintained as a PDP-11 disk file. The maintenance panel hardware/software package is used primarily for simulating the Input Signal Conditioner (ISC). This involves generating the equivalent of an ISC interrupt, which drives the software system, and filling the ISC buffers with synthetic data.

In the LAMPS system the SCU functions as system controller and post processor. The testing, therefore, proceeds along two lines. One is to check that the communication protocols involved in system control function as described in the Interface Document [6]. Using the SIO Adapter emulation capability on the PDP-11, streams of SCU type commands have been transmitted to the ASP and the various acknowledgements checked. The other line of testing has been to check the actual signal processing by loading synthetic input and examining the post processor (PDP-11) output. This procedure was used for Bathythermographic (BT) processing [9]. The BT buoy transmits a pure sinusoidal signal whose frequency varies linearly with temperature. The LAMPS software determines the frequency over certain time intervals by counting zero crossings. It then computes the temperature for each interval according to the linear formula. It is a computationally simple processing mode, but it was a good vehicle for putting the whole test procedure in order.

Similar testing is being conducted on the Ambient Noise Measurement (ANM) Processing. This uses a more conventional spectral analysis procedure to determine the background noise in various frequency bands. Pure sinusoids as used in BT seem to cause concentrated noise in the appropriate band, as would be expected. The testing will probably involve checking the correspondence between input and output amplitudes and using signals of mixed frequencies.

Thus, as a result of these initial efforts on the LAMPS software, the usefulness of the various facilities which the test bed offers has been demonstrated.

CONCLUSIONS

It was desired to introduce the AN/UYS-1 militarized signal processor into a laboratory environment for use as a software development and verification tool. A test bed was devised using a PDP-11/45 general purpose minicomputer as a host. In order to simplify and expedite access to the micro-level and assembly-level machine, two interfaces consisting of both custom and off-the-shelf hardware as well as comprehensive support software packages were designed, implemented, and tested at NUSC. The test bed development, now complete, has been exercised by various demonstration scenarios in an effort to debug its various hardware and software parts and provide evidence of its capabilities. At present, the facility is being utilized by the LAMPS program for software test and verification.

REFERENCES

1. "Advanced Signal Processor, Technical Description", IBM Corp., FSD, Manassas, Virginia 22110, July 1974.

2. MOLINO, F. W., "IBM Analyzer Unit (AN/UYS-1) - Maintenance Panel Port Interface", NUSC/NL Technical Memorandum No. 771165, 15 August 1979.

3. VERNAGLIA, J. E., "A DOS-MX Driver for the PROTEUS Maintenance Panel Interface", NUSC/NL Technical Memorandum No. 771185, 1 October 1977.

4. "PROTEUS Analyzer Unit, Preliminary Maintenance Manual", IBM Corp., FSD, Manassas, Virginia, 15 December 1975.

5. BERNECKY, W. R., "User's Guide for the Test Bed Command Language", NUSC/NL Technical Memorandum No. 771098, 27 May 1977.

6. "PROTEUS Interface Document", IBM Corp., FSD, Manassas, Virginia.

7. "Input/Output Interfaces, Standard Digital Data, Navy Systems", MIL-STD-1397 (SHIPS), 30 August 1973.

8. VERNAGLIA, J. E., "Support Software for the PROTEUS Digital Channel/NTDS/UNIBUS Interface", NUSC/NL Technical Memorandum No. 781228, 18 December 1978.

9. VERNAGLIA, J. E., "LAMPS Bathythermographic (BT) Processing; testing of", NUSC/NL Memo Ser: 9383-95, 27 July 1979.

DISCUSSION

CAPT C.M. Rigsbee Why didn't you use the NTDS channel I/O on the ASP instead of developing the NTDS interface box?

F.W. Molino The ASP Advanced Development Model, which NUSC has, was available only with Proteus Digital Channel I/O; NTDS channels became available in the production version. An NTDS channel interface for PDP 11 minicomputers was designed and built by NUSC. It has fairly general use, since many U.S. Navy militarized peripherals and computers have this standard I/O channel. The hardware to convert from NTDS to PDC protocol was not a new development, it had previously been designed by G.E., Syracuse, New York, for use in another application.

R. Seynaeve What percent of the ASP programming is done in:-

(a) Microcode

(b) Using a block language or a standard set of signal processing functions?

F.W. Molino Currently:

Arithmetic Processor (AP) : 100% microcode

Control Processor (CP) : 99% assembly
1% high level (SPL/I)

Future (pre-ACOS):

CP : 80% high level, 20% assembly

Future (post-ACOS):

CP : 50% block language, 50% high level.

Y.S. Wu How big an effort?

F.W. Molino Two people, three years. The majority of the design, implementation and debug efforts were performed by two people over a period of about three years.

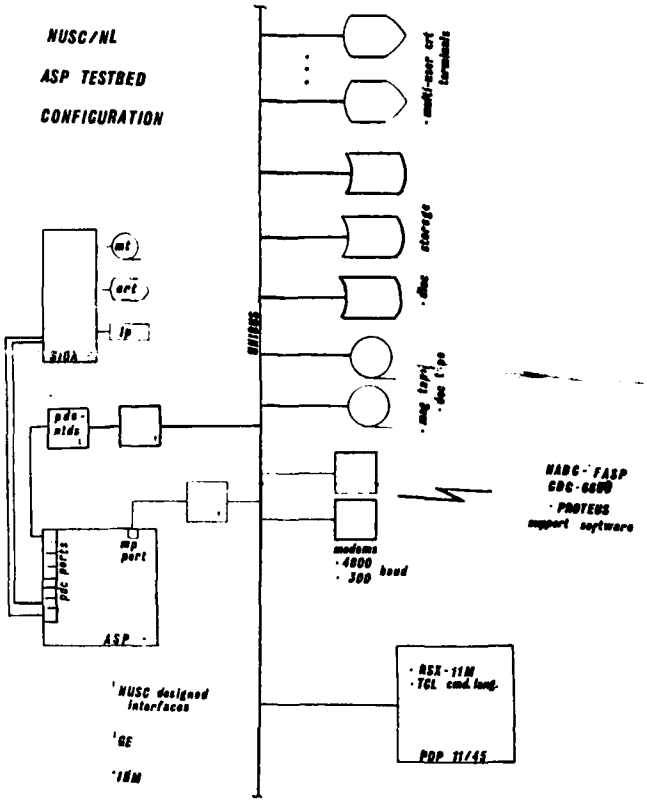


FIG. 1

AN/UYS-1 FUNCTIONAL BLOCK DIAGRAM

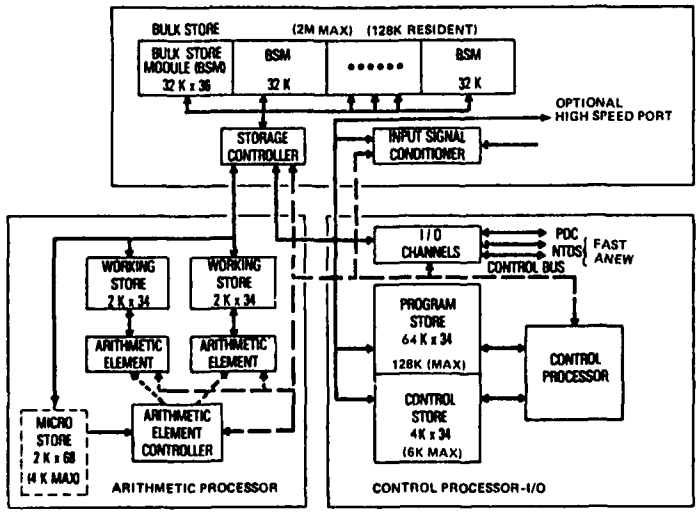
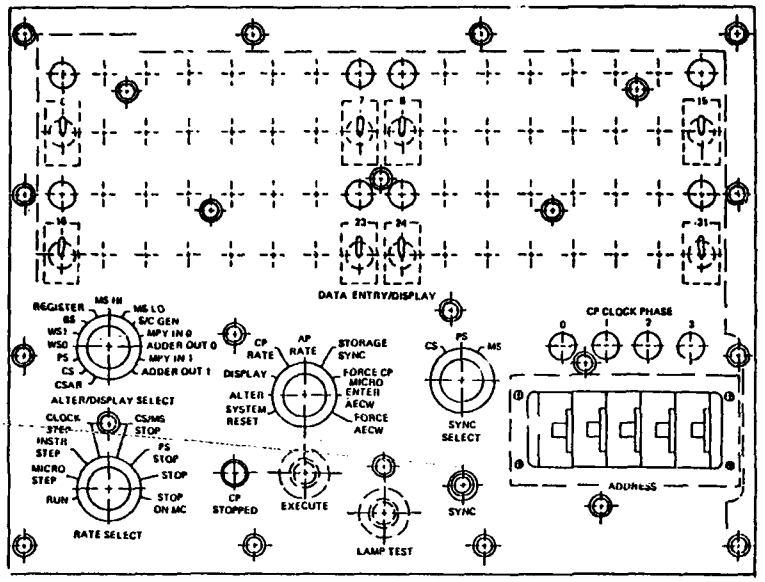


FIG. 2



Maintenance Panel Component Layout

FIG. 3

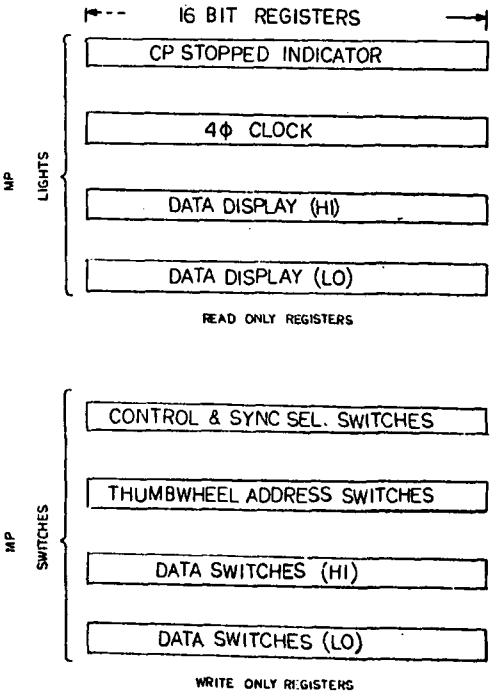


FIG. 4

MAINTENANCE PANEL INTERFACE
REGISTER DEFINITION

PDC INTERFACE CONFIGURATION

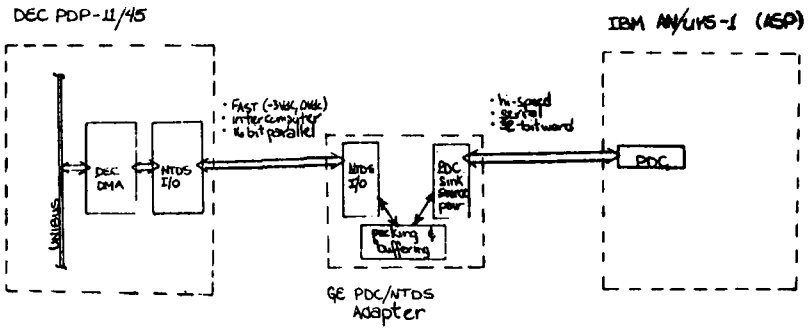


FIG. 5

MASCOT — A MODULAR SOFTWARE CONSTRUCTION SYSTEM

by

Donald Nairn
Admiralty Underwater Weapons Establishment
Portland, Dorset, U.K.

Mr D. Nairn gave an informal presentation on the MASCOT system (Modular Approach to Software Construction, Operation and Testing), which he had previously mentioned in his presentation "Development of Military Argus Computers and MOD Bus for Close-coupled Signal Processing Applications" (Paper 11 of these Proceedings). The basic characteristics of MASCOT had been described in this paper, and the informal presentation was given in response to the interest shown by the audience.

The presentation was supported by the following Vu-graphs; these should be taken in conjunction with the author's previous paper. (Ed.)

MASCOT - MODULAR SOFTWARE METHODOLOGY

- SYSTEM DESIGNER DESCRIBES SOFTWARE ON A SINGLE ACP DIAGRAM
- SYSTEM DESIGNER SPECIFIES ACTIVITIES AS INDEPENDENT SOFTWARE MODULES TO BE WRITTEN SEPARATELY
- SYSTEM DESIGNER WRITES ALL MODULE INTERCONNECTION AND SYNCHRONISATION SOFTWARE HIMSELF
- SYSTEM IS DATA-PACKET DRIVEN RATHER THAN CLOCK INTERRUPT DRIVEN
- MASCOT PROVIDES A SPECIAL SUPERVISOR TO RUN THE FINAL SOFTWARE SYSTEM

MASCOT - SOFTWARE METHODOLOGY

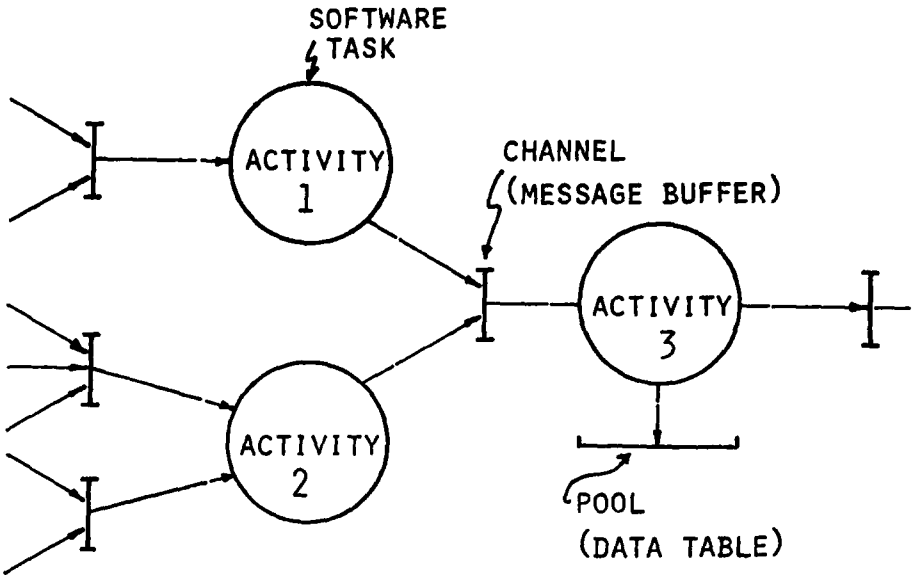
MODULAR APPROACH TO SOFTWARE CONSTRUCTION, OPERATION AND TEST

(EXAMPLE - SAY SYSTEM DESIGNER AND 6 PROGRAMMERS)

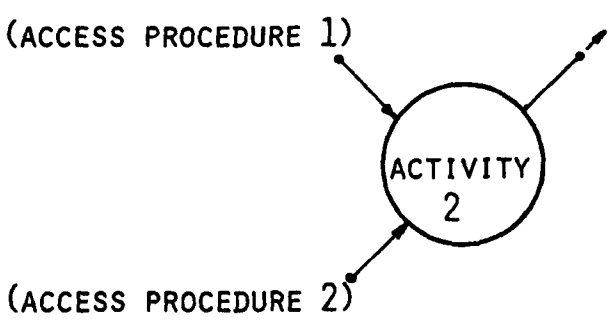
SYSTEM DESIGNER HAS TO:

1. DESIGN OVERALL SYSTEM - ACP DIAGRAM
   - ACCESS PROCEDURES
2. SPECIFY AS A SERIES OF SEPARATE TASKS (ACTIVITIES) TO BE GIVEN TO INDIVIDUAL PROGRAMMERS.
3. "STITCH" THE SEPARATELY WRITTEN TASKS TOGETHER TO FORM THE FINAL SYSTEM.
4. DEBUG THE OVERALL SYSTEM.

(D I V I D E AND R U L E)



PROGRAMMER's EYE VIEW:



ACTIVITIES REALLY
SEPARATE THREADS
EXECUTED IN PARALLEL

(- SYSTEM INDEPENDENT)

ACCESS PROCEDURES

(WRITTEN BY SYSTEM DESIGNER)

- MESSAGE TRANSPORT BETWEEN ACTIVITIES
- SYSTEM SYNCHRONISATION
  SYSTEM IS "DATA PACKET" DRIVEN

PERHAPS JUST TWO GENERAL PURPOSE PROCEDURES:-

GET $(p_1, p_2 \ldots p_n)$ (MESSAGE i/p)

PUT $(p_1, p_2 \ldots p_n)$ (MESSAGE o/p)

SYSTEM SYNCHRONISATION

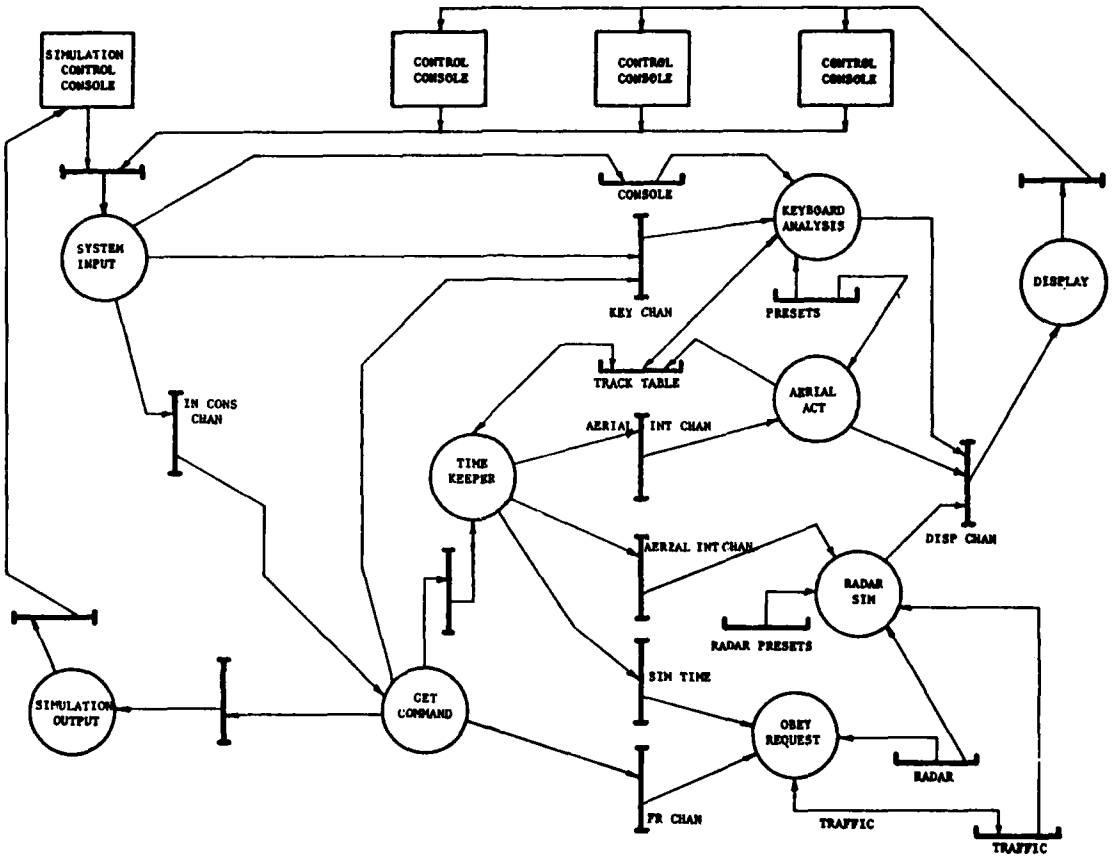
1. ACCESS PROCEDURE MUST "CLAIM" A CHANNEL BEFORE USE AND "RELEASE" THE CHANNEL AFTER USE.
2. IF CHANNEL ALREADY ENGAGED, ACTIVITY AUTOMATICALLY SUSPENDS ON CONTROL QUEUE OF THAT CHANNEL.
3. WHEN THAT CHANNEL BECOMES FREE, ACTIVITY AT FRONT OF ITS CONTROL QUEUE (IF ANY) IS AUTOMATICALLY RE-ACTIVATED.

SYSTEM OPERATION AND TESTING

MASCOT PROVIDES:

1. SYSTEM BUILD SOFTWARE COMPILE-TIME CHECKS.
2. RUN-TIME SUPERVISOR KERNEL.
3. MONITOR AND DEBUG AIDS
   - REAL TIME LOG
   - SPECIAL KERNELS FOR RUN-TIME CHECKS.

AN ACP DIAGRAM



PANEL ON SOFTWARE

Chairman: CAPT C.M. Rigsbee

Members : E. Hug
P.A. Rigsbee
R. Seynaeve
S. Weinstein
Y.S. Wu

CAPT C.M. Rigsbee I wish to give a little background on SPL/I. Prior to the SPL/I Project we had severe cost over-runs, particularly in the maintenance of completed systems, usually because each system had its own language. It was, therefore, decided to force contractors to use a standard language, and to develop software on a Navy computer via terminals. Since major changes had to be made to the fleet software, the modular approach was used, otherwise it would have been hard to make even minor changes. We own the software and SPL/I allows for close monitoring of the contractor. For me this is the main justification, not the desire to use high-level languages; software can be used to specify hardware requirements, in effect, we will buy any hardware that will run this software. Testing, another problem, can be left to the contractor to do. It is now being realized, however, that software can be as much as a pandora box as hardware, in fact, hardware is probably more predictable, and a software criteria on reliability and projected cost is being sought.

R. Seynaeve I should like to comment on block languages, i.e., languages made of signal processing primitives. I personally feel that the block-language concept is still undervalued in signal processing. Here at SACLANTCEN we have developed and used ITSA, a block language signal processing system, for over nine years and, although ITSA is in no way perfect (it has never been fully completed), we find it very appropriate in a research environment and we are going the same way with our array processor based systems. In practice, I feel programs are much easier and much more error-free with block language. It is also very good for run-time checking and for transportability. I believe intuitively that a block language approach will be more easily adaptable to distributed processor architecture than other approaches. I also found that a somewhat comparable approach consisting of enhanced FTN plus a set of primitives, in practice ends-up with two languages instead of one to deal with. A signal processing block language must be a total language and should include I/O and display primitives. I expect that in about six months our users will be using such a block language with our array processors.

CAPT C.M. Rigsbee ACOS is an attempt at a block language, integrated with SPL/I partly for political reasons. It is hard to develop a block language from scratch in DOD.

Y.S. Wu There is currently no new language under development in DOD; we are waiting for ADA.

D. Nairn Suppose you had used the worst of the DOD languages. Would it have been an advantage, is any standard better than none?

CAPT C.M. Rigsbee This was our problem; we would have had to extend CMS-2 or else continue the development of SPL/I. It was decided purely on cost-effective grounds that a new language was needed. ADA has a lot of SPL/I in it. Perhaps block programming will help with reliability.

R. Seynaeve Yes, it should.

Y.S. Wu That is why I am enthusiastic, and it should enforce more discipline on programmers.

D. Nairn U.K. decided that we should stay with CORAL for the next few years, even though it is showing its age.

CAPT C.M. Rigsbee CMS-2 was like that, but it was finally decided to change.

H. Urban What happened to the U.S. Navy architecture standardization, or PDP-11?

Y.S. Wu We selected the PDP-11 design, then the Army took over and tried to obtain a licence from DEC for the bus design, but they wouldn't cooperate.

D. Nairn They were not willing to respond to a similar approach by the U.K. The U.K. thinks that transportability is guaranteed by having the same instruction set, some languages don't guarantee it.

Y.S. Wu We standardize on UYK-20 set, then the language standard is added on top.

D. Nairn Does ADA have an instruction set defined?

Y.S. Wu No.

D. Nairn What then?

CAPT C.M. RIGSBEE ADA is not yet complete.

Y.S. Wu ARPA normally would do a project but has no experience with delivering it to the fleet. Until DOD makes a decision we just don't know. The language is the easy part, the support and running environment is the hard part.

D. Nairn CORAL proved expensive to "police" so that compiler implementation could be certified for government use. MASCOT will not be policed in the same way because of expense. A user/supplier group will control the standard.

Y.S. Wu ADA has some parts done in-house so that contractors can't pirate our compilers, they must use my code generator and operating system. Unless you do something that drastic you are in trouble. There are no SPL/I pirate editions.

CAPT C.M. Rigsbee We forced IBM to develop programs on our CDC 6600 via a terminal, and automatically monitored development.

Y.S. Wu IBM wanted to write everything in 360/10 assembler, but we refused and insisted that FTN be used. Our justification being that all Navy laboratories had different computers and therefore the software must be transportable.

CAPT C.M. Rigsbee Take maintenance contracts — anyone can maintain software on our machine, or via terminals, or on their own machine. This is not possible, if one contractor has made everything in assembler. This way we can ask for quotations from a wide range of companies.

Y.S. Wu We never release the source code. Once we had to encrypt access to our machine — even CAPT Rigsbee's people couldn't get on.

CAPT C.M. Rigsbee Any other point?

Y.S. Wu Thinking on a standard set of primitives — look how many FFT's there are about.

F.W. Molino In fact, there are six on DOD systems.

CAPT C.M. Rigsbee Yes, with block languages standardization it will be necessary from the very beginning.

D. Nairn What about changes to the definitions of the primitives, should this become necessary or perhaps even desirable?

CAPT C.M. Rigsbee One must carefully evaluate the effects a change will have.

R. Seynaeve I have the impression that a block language makes such assessment easier.

Y.S. Wu That is why I am so enthusiastic.

R. Seynaeve When will the U.S. standard primitives be chosen?

Y.S. Wu In a year — 1980/81.

SESSION V BEAMFORMING

REAL-TIME EXPANDED-BAND BEAMFORMING USING A COMPLEX HETERODYNER AND FAST ARRAY PROCESSOR

by

D. Vance Crowe
Defense Research Establishment Atlantic
Dartmouth, Nova Scotia, Canada

Introduction

At the Defense Research Establishment Atlantic in Dartmouth, Nova Scotia, Canada a data aquisition sytem has been engineered to monitor and beamform data from 30-element line arrays. This system has been designed to work with either towed or free floating arrays deployed from the Canadian research vessel Quest as depicted in Figure 1.

Hydrophone signals are converted to a pulse code modulated (PCM) bit stream which is transmitted or cabled to Quest for recording and analysis. The PCM signal is recorded directly on an analog tape recorder using high density recording (HDR) techniques.

The digital data are fed directly to a minicomputer system for analysis of the entire bandwidth of the hydrophone signals, originally sampled at 2000Hz. When CW experiments are performed, the data are preprocessed using a multi-channel digital filter-translator. The signals are complex heterodyned, low pass filtered and decimated. In this way, narrow band analysis of the CW signals is accomplished using the complex signal fed to the computer.

1. Hardware

A quick overview of DREA's digital data acquisition hardware is shown in Figure 2. The analog conversion to a digital data format for transmission to Quest is done underwater inside the lower electronics unit. Once onboard the 640 kilobit per second data are recorded on a High Density Recorder. Several types of analysis are done using the live or recorded data. Audio monitoring is accomplished using the reconstructed analog signals. The digital inputs

are fed directly into our computer system for processing of the array's hydrophone signals.

This system is the result of an evolution process rather than one single design step. Commercially available hardware was used where cost and technical performance permitted. The underwater electronics packaging was built at DREA and makes extensive use of CMOS integrated circuits. The LEU, lower electronics unit, package is small, light and can be supplied with power from batteries for periods of 24-48 hours.

Data from the array arrive as a 640 KHz PCM signal composed of 32 words of 10 bits each. One word is a synchronization code. This is followed by data words from the 30 hydrophones. The last word in the sequence is reserved for multiplexed nonacoustic sensor data: three depth sensors, battery voltage sensors, and a compass. The entire PCM bit stream is randomized and recorded directly on one track of a High Density Recorder. At DREA, seven passes are made on a 14 track tape to record seven hours of data. More tracks would be used but the tape wear and stretching generally has been found unacceptable if extensive analysis is performed with any one tape. This often happens when an unusual event occurs or an underwater system has had an unexpectedly short life.

The serial code is reformatted to parallel words with a PCM reconstruction unit made by Decommutation Systems Inc., model 7101. This programmable unit extracts the synchronization code and feeds the 10 bit words to several devices: an analog reconstruction unit made at DREA, a nonacoustic sensor data logging station, a digital filter, and a minicomputer.

The digital filter is a commercial unit made by MacDonald Dettwiler and Associates, Vancouver (MDA). This unit can be programmed with two 512- or four 256-coefficent digital FIR filters. In its real mode, the digital words from each hydrophone signal are routed through a low pass filter. As well the sampling rate is reduced by a factor related to the designed filter cutoff frequency. For this experiment, decimation factors of 25 and 42 are selected from the front panel.

The MDA filter, however, is most often used in complex mode. Each data channel is first multipled by the sine and cosine terms of a heterodyning frequency. The resulting complex word is then low pass filtered and the

time series decimated. The output complex signal represents a shifted narrow band of the original input signal. The band centre is the heterodyne frequency and is selected at integer multiples of the original sampling frequency divided by 512. The selected low pass filter determines both the bandwidth and the decimation factor.

The Underwater Acoustics Division is attempting to standardize digital connections to provide a common interface to data processing equipment. This commonality and the flexibility of the MDA filter-decimator have meant that the filter has been used in all sea trials since it was interfaced to the computer in April, 1979.

The analysis is controlled and centered around a PDP11/34 minicomputer. Peripherals include three 2.5 Megabyte disks, two Kennedy tape transports capable of 200 Megabytes per second transfer rate, 64K words of MOS memory, a Versatek printer-plotter, a Tektronix graphics terminal with hard copy and a CRT alphanumeric terminal.

Inputs to the PDP 11/34 computer from the MDA filter or the decommutation system are through direct memory access ports. These ports are Digital's DR11-B's with the addition of one circuit board containing a first-in-first-out (FIFO) memory. This 64 word FIFO allows delays in software and hardware processing to occur before data is lost. This simple hardware addition has simplified the design of real-time systems. As an example, maximum interrupt service times of 20 microseconds meant that one program used triple buffering, disabled the operating system clock and blocked keyboard interrupts. After the FIFO modification, interrupt service times went up to 1.28 milliseconds and all of these drastic steps were eliminated.

This 11/34 system hosts an AP120B, an array processor built by Floating Point Systems. The AP120B is a multi-bus microprogrammable arithmetic unit. Main data memory consists of 128 kilo-words of 38 bit floating point data. Arithmetic functions use a 500 nanosecond floating point multiplier and a 333 nanosecond floating point adder. A separate 16 bit arithmetic logic unit handles address calculations. The 64 bit microcode contained in the 512 word program source memory is capable of controlling all arithmetic functions and most of the data paths simultaneously. By carefully writing optimized programs, computationaly intensive processing can be done up to one hundred times faster than with the 11/34 minicomputer.

2. Software

Software binds all of this high speed processing equipment together. The AP120B has spent hundreds of hours analyzing data using data prerecorded on computer tape. All analysis of the data was done with programs written in Fortran, using libraries supplied with the AP120B, and the RT-11 operating system to handle the input and output functions. During this first year of operation, speed of processing was often sacrificed to ease and speed the efforts of the scientist in designing, coding and debugging his processing routines.

The addition of the heterodyning filter led to a request to beamform in real-time the narrow band data for upcoming sea trials in September 1979. Existing processing software library routines and features of the RT-11 operating system have proven to be flexible and fast enough to handle the output sampling rate from the MDA filter without resorting to extensive use of specialized software written in assembly language.

The real time processing overlaps the time series data by 50% for statistical purposes. By combining new samples from each hydrophone with the previous 256 samples, a 512 word data record is formed. The FFT's are done on the 30 acoustic channels. The power spectra from each hydrophone signal are averaged separately and saved within the array processor. Thus the system provides a monitoring facility of the acoustic performance of each element in the array.

The second task of the array processor is to compute the cross-spectra of all combinations of elements. The cross-spectra from combinations that have the same physical distance between elements are accumulated together. Averaging of cross-spectra according to inter-element spacings is one step in cross-spectral beamforming and is discussed in Appendix A.

Digital Equipment's RT-11, version 3, is the host operating system used for DREA's real time processing requirements. RT-11 is preferable to multi-task operating systems. It is small in size, is reasonably fast with I/O system calls, and is totally passive in turning over control to the user with exacting time requirements. A disadvantage is the small 28K memory space that RT-11 supports for programming. About 6K is used for the operating system.

One feature of RT-11 is that it can run a real-time job in the foreground with priority over a background job. Control remains with the foreground job as long as it is active. This job can suspend itself while waiting for an external event to occur: such as new data becoming ready for processing. The background program runs until the foreground job is resumed by this external event.

The real-time beamforming program is organized with the data input and processing handled in the foreground. This processing is done continuously and the data are exponentially averaged. A fraction of the averaged spectra is kept and the difference is made up by the new spectral sample added to the average. The effect is that up-to-date spectra and cross-spectra for all hydrophones are available on a continuous basis. The background programs service the often volatile operator who can quickly switch displays from one function, or format, to another without waiting for the accumulation of new data. One processing program is able to replace several analysis programs by using the foreground-background features of RT-11. A selection of display programs is still needed but the processing is not built into them.

In addition to beamforming, these data had to be recorded on computer tape in real-time. Part of the foreground processing job is to write the data on magnetic tape after each new block of data is collected. A special assembly language routine senses the loading of a new magnetic tape and writes the necessary volume, file header and data labels. This makes the tapes compatible with our operating system magnetic tape handler and analysis programs. Once initialized, the tape recording proceeds without the loss of any data until the data file is closed. Files are closed if the end of the tape is reached or the operator keys a close command.

2.1 Software Objectives

The flow chart of the foreground program is detailed in Figure 3. The design of the foreground program when coupled with the background display accomplishes several objectives. Major items and features included in the program to meet these objectives are presented below.

1. A small compact assembly lanaguage program inputs raw

data directly into upper core and controls transfers to the magnetic tape and the AP120B.

2. Magnetic tape data files are written which are compatible with ANSII standard tape labels. No data are missed while recording at sampling rates of up to 64 kilohertz.

3. The processing algorithm uses the array processor's supplied executive and library functions refered to as APEX. Process algorithms and APEX are written in Fortran.

4. A communication link to background programs handles several types and formats of data. This flexibility meets the needs of a multiplicity of displays.

5. Display programs can be easily updated without any restructuring of the underlying data or processing code.

The following features are incorporated in the foreground program to provide flexibility and good interaction with the operator or scientist.

6. The logging of the magnetic tapes used, the file names and the count of magnetic tape parity errors is done on hard copy.

7. The operator is able to change sampling frequencies and heterodyning frequencies without restarting the program.

8. The operator can select channels to be omitted in the beamforming algorithm if a hydrophone becomes noisy or malfunctions.

9. The operator, or the background program, can suspend parts of, or all of, the processing section of the foreground program to gain more time for background processing and displays.

2.2 Software Performance

The foreground part of the real-time beamforming program fits within 7000 words of memory. Approximately 15,000 words are left for the display portion of the real-time beamforming and display program. The division of foreground space according to function is shown in Table I. Most of the program space is devoted to system routines and to the processor coding which is written in Fortran.

The real-time beamforming program is fast enough to handle a 50Hz sampling rate from the MDA filter. When comparing this program with other processors, this rate might be compared to 6000 real samples per second as the data rate is overlapped by 50% and includes 30 complex data channels at each sampling interval.

The major cost of this system is primarily in the software. This cost is proportional to the number of lines of code whether those lines are in a high level language or assembler. At present, about a hundred lines of Fortran code control the AP120B and access by the background to the data base. Table I shows that this is a small fraction of the total number of lines of code generated for this program. This program would run faster and use less memory space by increasing the use of assembly language programming within the 11/34 and micro-coding within the AP120B but the cost would be great. Another advantage of maintaining the processing algorithm in Fortran is that process modifications are easily introduced.

One more conclusion can be inferred from Table I that is common to a lot of data processing problems. The bulk of the effort in programming is spent designing how to get the data into the machine and then how to get the processed data out again. But now that this input/output structure has been formalized, optimization for processing speed can be introduced quite easily. Two things will have to happen before efforts in optimizing begin: the completion of the display programs and a shakedown of the system that only a sea cruise seems to provide.

Graphic display software, with its requirements for attention to detail, consumes a large proportion of the resouces of the programmer and the 11/34. About 15,000 words of memory are available for the background display program. When programming graphics in Fortran, this space is very quickly filled. The simple display program used here by way of example occupies 11,000 words.

Figure 4 shows a sample of the display which is created while the foreground program is running in real time. The power spectra of 30 hydrophone signals are displayed. Frequency is across the X axis and the ordinate is in decibels. Each spectrum is offset by 10dB. Notice that channels 15, 16 and 30 are noisy. Indeed, it is one of the purposes of this display to assess quantitatively the acoustic performance of the hydrophones.

Conclusion

A computer, an array processor and a special purpose digital filter have been connected to DREA's digital data acquisition system. This system is fast and flexible enough to handle a variety of analysis problems with live and recorded data, using available software. Hundreds of hours of completed analysis have proven this configuration to be an extremely reliable analysis system.

A program for beamforming line array data in real-time over a narrow frequency band has been developed which has many advantages over previous efforts at real time analysis. The raw data are recorded concurrently with the real-time beamforming so that more detailed analysis can be done later with other computer systems. The display and processing algorithms are written in Fortran providing a flexibility that has not been available with previous algorithms written in assembly langauges. Data are analyzed continuously and exponentially averaged. Portions of that data can be selected for an immediate display on the acoustic performance of the underwater array as a whole or as individual elements.

Minicomputers and digital hardwired processing systems have been going to sea for some time. Previously, at DREA, assembly language programs had to be written to meet the speed requirements of sonar applications. These results were often inflexible, poorly adaptable to changes and of limited utility. In contrast, a programmable array processor with a library of microcoded algorithms, such as the AP120B, permits a scientist to use a high level language to build the real-time processing for new underwater research arrays. The decrease in time and energy required to provide future experiments with real-time processing will be significant now that the software structures have been designed and tested.

Acknowledgement

Special thanks are due Dr. John H. Stockhausen, leader of the horizontal line array team at DREA. His contributions and encouragement in the development of this program and paper are most appreciated.

TABLE I

	Words of Memory	Lines of Code
Mainline Control	492	266 (macro)
APEX + Processing	2096	100 (Fortran)+ 130 (macro)
Mag Tape Handler	1321	728 (macro)
Terminal Handler	584	100 (macro)
Backgnd Connection	91	50 (macro)
System & Fortran	2754	n/a
Totals	7849	1374

REFERENCES

1. R. D. Arnott & J. D. Markell, "Fortran Control of Real-Time Signal Processing with High-Speed Processors", IEEE Transactions on Acoustics, Speech, & Signal Processing, vol. ASSP-26, pp.278-284, Aug. 1978

2. N. Yen, "Ambient-sea-noise Directionality: Measurement and Processing", Journal of Acoustical Society of America, vol.62, pp.1176-88, Nov.1977.

3. H. P. Bucker, "High-resolution Cross-sensor Beamforming for a Uniform Line Array", Journal of Acoustical Society of America, vol.63, pp.420-4, Feb, 1978.

4. J. R. Williams, "Fast Beam-forming Algorithm", Journal of Acoustical Society of America, vol.44, pp.1454-5

DISCUSSION

R. Seynaeve Regarding the mentioned digital filter, could you give information on:

(a) Cost

(b) Size

(c) Address of supplier

(d) Memory size of AP120B and cost of AP120B system?

D.V. Crowe (a) Present cost of MDA filter is Canadian $24,000.

(b) Physical size is 10 cm height in rack; a fuller description on speed, number of channels etc., will be provided [see attached Circuit Description].

(c) The suppliers are:
MacDonald,Detwiller & Associates Ltd.
10280 Shellbridge Way
Richmond, British Colombia, Canada.

(d) 128K of 38-bit main data memory drove the price of the AP120B system to $100,000.

CIRCUIT DESCRIPTION — D.V.Crowe

The non-recursive digital filter/translator, model 175 NDF-T, performs the operation of discrete convolution (spectral shaping) by multiplying each data word with a certain number of coefficients, a coefficient subset. The size L of a subset is determined by

(1) the total number of coefficients N forming one filter, and

(2) the subsampling factor M:

$$L = N/M$$

M subsets form a coefficient set which defines one filter.

If frequency translation is required, the data word has first to be multiplied with a cosine and sine value, generated by the frequency synthesizer. The two resulting products will then be multiplied with the filter coefficients.

The 175 NDF-T accepts data words from different channels, time multiplexed into frames. A frame will consist of one data sample from each channel, equally spaced and consistently ordered in time. The number of input channels permitted can be calculated from the filter design parameters N and M:

$$CHmax = \frac{1024.M}{N} \quad \text{in the real mode}$$

$$CHmax = \frac{512.M}{N} \quad \text{in the complex mode.}$$

The maximum input data rate or single channel throughput of the filter is also a function of the parameters N and M. in the real mode of operation, the minimum sampling period is given by

$$Treal = \frac{N}{2.5\ M} \quad \text{(microseconds).}$$

If more than one channel is used, the minimum period between samples of the same channel is given by Treal times the number of channels in effect.

In the complex mode of operation, two output channels are generated and filtered for each input channel. Furthermore, the translation operation requires one multiplication per output channel. Hence, the minimum sampling period is then given by

$$Tcmplx = \frac{N + M}{1.25 \cdot M} \quad \text{(microseconds).}$$

The same increase as in the real mode has to be made for multi-channel operation.

If subsampling is implemented ($M > 1$), data is not output for each frame. Subsampling by a factor M means that only each M frames, one (in the complex mode two) data words per channel are output. M frames form one cycle.

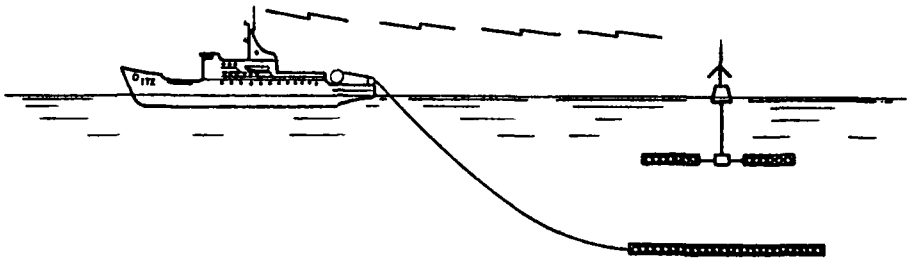


FIG. 1 QUEST AT SEA WITH BOTH A TOWED UNDERWATER ARRAY AND A FLOATING ARRAY. DATA AT 640 KILO-BITS/SECOND ARE CARRIED OVER A UHF RADIO LINK OR THE TOW CABLE.

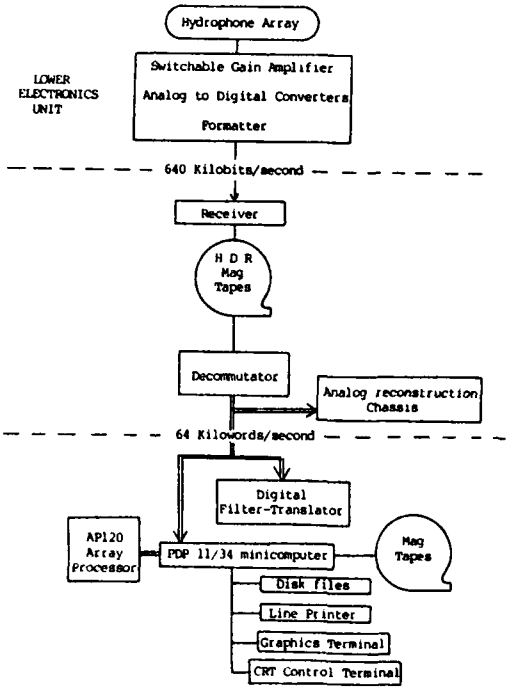


FIG. 2 EQUIPMENT BLOCK DIAGRAM SHOWING THE UNDERWATER ELECTRONICS, THE SURFACE RECORDING SYSTEM, AND THE COMPUTER ANALYSIS EQUIPMENT.

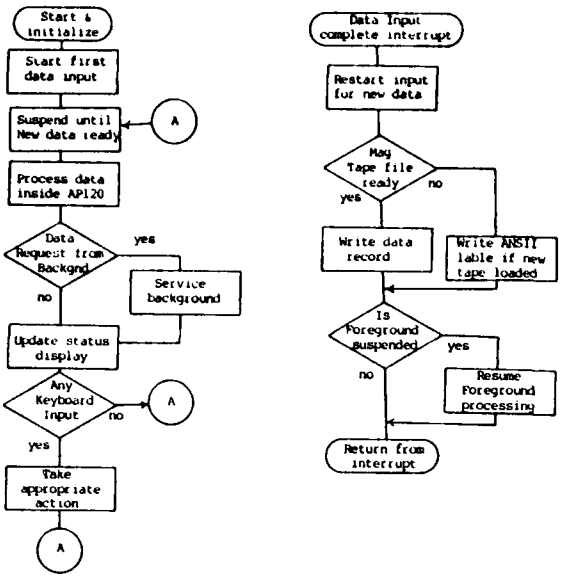


FIG. 3 FLOW CHART OF THE FOREGROUND ANALYSIS PROGRAM. DISPLAY PROGRAMS USING THIS DATA ARE RUN AS TIM IS AVAILABLE IN THE BACKGROUND.

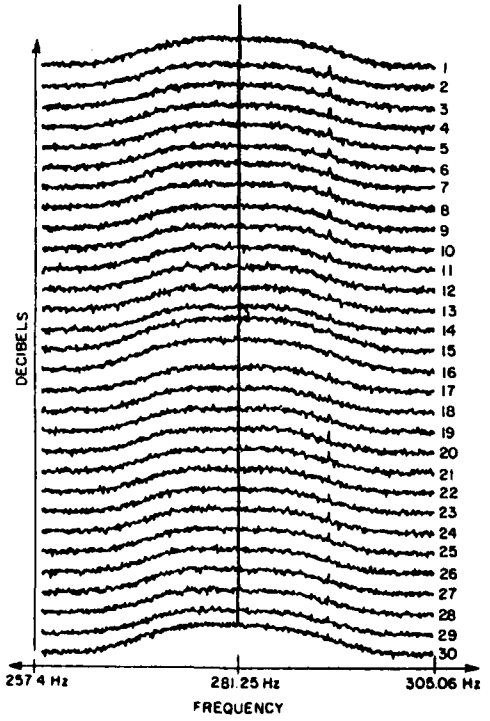
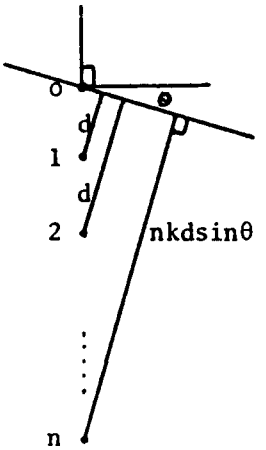


FIG. 4 NARROW-BAND SPECTRA FROM ALL 30 ELEMENTS WITHIN THE ARRAY. SPECTRA ARE OFFSET BY 10 dB.

APPENDIX A: Beamforming with a Uniformly Spaced Line Array



Beamforming to angle, θ, is accomplished in frequency domain by summing over all elements the phase-delayed signals at each hydrophone. At frequency, f, this is given by[2,3]

$$B(f,\theta) = \sum_{n=o}^{N-1} S_n(f) e^{-jnkdsin\theta}$$

where B is the beam signal of f and ;
N is the number of elements, which are numbered 0 to N-1;
$S_n(f)$ is the signal at element n, and at frequency f;
k is 2π/wavelength;
d is the inter-element spacing;

and nkdsin θ is the path difference, in wavelengths between the 0^{th} and the n^{th} element to the plane wavefront arriving at .

The beam power becomes

$$B \cdot B^*(f,\theta) = \sum_{n=o}^{N-1} \sum_{m=o}^{N-1} S_n(f) \cdot S_m^*(f) e^{-j(n-m)kdsin\theta}$$

The $S_n \cdot S_m^*$ term is the cross-spectrum for elements n and m. For the uniform line array, cross spectra between elements of equal spacing can be summed together, so that the beampower can be written as

$$B \cdot B^*(f,\theta) = \sum_{\ell=-N+1}^{N-1} \left(\sum_{n=o}^{N-|\ell|-1} S_{n-\frac{\ell}{2}+|\frac{\ell}{2}|} \cdot S^*_{n+\frac{\ell}{2}-|\frac{\ell}{2}|} \right) \cdot e^{+j\ell kdsin\theta}$$

The above equation is an inverse discrete Fourier transform. For this the FFT algorithm[4] can be used if beams are spaced at integer multiples of kdsin θ/NB. NB is the number of beams calculated by adding NB-N zeros in the cross-spectral domain.

SONAR BEAM-FORMING WITH AN ARRAY PROCESSOR IN REAL TIME

by

Walter G. Wagner

KRUPP ATLAS-ELEKTRONIK

2800 Bremen, Germany

ABSTRACT A real-time sonar beam-forming system is described which consists of a minicomputer as host computer and an array processor. Beam-forming is accomplished by using the FFT algorithm in the processor.

The system is also connected to a multi-channel PCM magnetic tape recorder. Processing can be done either on-line or off-line. During sea trials the entire equipment is installed on board a research vessel.

An acoustic towed array supplies up to 40 input signals to the beam-former. The AP 120 B from Floating Point Systems is used as the array processor.

INTRODUCTION

Up until now, real-time beam-formers have usually consisted purely of hardware. The inadequate throughput of standard minicomputers prevented their us. Software solutions are made possible only by employing an array processor, which is specially suitable for processing large arrays of data. In contrast to the general purpose computer, an array processor has several arithmetic units and a "pipelined" structure. This allows operations on data arrays to be carried out much faster than the corresponding individual operations. In particular, address calculations and memory access are separated from the arithmetical computation and run in parallel with it. An array processor can thus work several hundred times faster than a typi-

cal general purpose computer.

In the following a report will be given on the realisation of a beam-former with the aid of an array processor; the following boundary conditions must be fulfilled:

- The sound signals are received with a line array having N equidistant hydrophones or hydrophone groups.
- The incident signal wave-front is plane.
- The unwanted noise signals at the individual hydrophones are mutually uncorrelated.

Under these conditions, the time-delay beam-former is the ideal detector. However, it is not an easy task to realise this in a digital computer. For a given sampling rate there is only a discrete number of exact beam directions. These are also called "natural" or "synchronous" beams [1]. There must therefore be a significantly higher sampling rate as required due to the sampling theorem, or use must be made of an interpolation process.

For realisation in the array processor, therefore, a process is used in which the delays are replaced by phase rotations [2, 3, 4].

In the following, a beam-former will carry out both beam-forming and beam-scanning.

1. Overview of the Hardware

The overall beam-forming system consists of

- o the acoustic towed array
- o the data recording equipment, and
- o the computer system for on-line beam-forming.

Fig. 1 shows the scheme of the data recording equipment consisting of

- o the PCM modulator
- o the magnetic tape recorder
- o The PCM demodulator.

The data can be taken from the output of the demodulator in analogue and digital form. The overall PCM equipment can process up to 40 individual channels. The digitalised data are fed directly to the array processor.

The array processor computes the directional pattern and passes the result on to the host computer. The host computer is employed merely to output data to appropriate devices.

2. Beam-Forming with FFT

2.1 Narrow-Band Signals

Fig. 2 is intended to clarify the notations which we will use.

Assuming narrow-band signals, it is possible to replace the delays τ_n required for beam scanning, by phase rotations $2\pi f_o \tau_n$ (f_o = centre frequency of the received signal). This can be done very easily with the aid of FFT (Fast Fourier Transform) [2, 3]. The representation of the received signal in complex form is required, which can be achieved for example by quadrature sampling.

This leads to the expression

$$(1) \qquad u(\vartheta) = \sum_{n=0}^{N-1} u_{on}(t)\, e^{j[\varphi_n(t) - 2\pi n \Delta x \frac{f_o}{c} \sin\vartheta]}$$

By simple rearrangement, we obtain

$$(2) \qquad u(\vartheta_m) = \sum_{n=0}^{N-1} u_n\, e^{-j2\pi \frac{mn}{N}}$$

where

$$u_n = u_{on}(t)\, e^{j\varphi_n(t)}$$

and

$$\vartheta_m = \arcsin \frac{m \cdot c}{N \cdot \Delta x \cdot f_o} \; .$$

$u(\vartheta)$ is also referred to as "angle spectrum" and $\frac{f_o}{c} \sin\vartheta$ as "spatial frequency". The increment between two angles ϑ_m and ϑ_{m+1} is in fact the angular resolving power of the antenna. For finer sampling of the angle range one simply has to fill up the series in (2) with zero elements. We thus obtain

$$(3) \qquad u(\vartheta_m) = \sum_{n=0}^{\tilde{N}-1} \tilde{u}_n\, e^{-j2\pi \frac{mn}{N}}$$

where $$\tilde{u}_n = \begin{cases} u_n & \text{for } n \leq N-1 \\ 0 & \text{for } n \geq N \end{cases}$$

The calculation of (3) is done with the aid of FFT. For this purpose $\tilde{N}$ must be chosen as a power of 2. It can be seen that we can thus process any number N of received signals where

$$N \leq \tilde{N} .$$

2.2 Broad-Band Signals

The principle of phase compensation can be applied only to narrow-band processes. At first sight it would therefore seem that it is no longer possible to process broad-band signals. However, if all N received signals are split up into a large number of narrow-band processes, then it is once again possible to use FFT, although a large number of band-pass filters is needed for this.

A much simpler solution is again offered by FFT, which can be performed very quickly in an array processor. The incoming signals are split up into time elements of length T_w and then filtered with the aid of FFT. A "spatial FFT" is then carried out for every frequency. In this way a directional pattern is obtained for every frequency band. One generally wants just one directional pattern. For this purpose the narrow-band angle spectra must be recombined. Simple summation cannot be used, since the relationship between space frequency or beam number and angle of incidence depends on the frequency. This fact is illustrated in Fig. 3 by means of an example. Summation must therefore take place along the lines ϑ = const.

Let us point out here a feature of this processing in the reception of noise signals. For linear antennas in general a definite directional correspondence is possible with monofrequent signals only if the distance between the hydrophone groups is less than half the wavelength. At high frequencies grating lobes occur. This limitation does not apply to noise signals. The state of affairs shown in Fig. 4 is thus obtained for the

FFT processing used here. Assuming that the hydrophones act as point objects, the space frequencies are repeated periodically in accordance with the spatial sampling theorem. The lines ϑ = const. are therefore also periodic. Above the frequency $f = \frac{c}{2\Delta x}$ portions of the neighbouring spatial frequency ranges are folded for the first time into the range under consideration. The summation of the narrow-band directional functions therefore has to take place on the lines

$$f = \frac{c}{\sin \vartheta}\left(f_x + k f_{xN}\right) \quad , \quad k = \ldots -2, -1, 0, 1, 2 \ldots \tag{4}$$

where f_x = spatial frequency

f_{xN} = spatial Nyquist frequency.

The length of the time windows T_w determines the loss of directional gain compared with the ideal time delay beam-former.

Simple considerations [3] lead to the following condition:

$$T_W \gg \frac{L}{c} \qquad L = \text{antenna length} \tag{5}$$

or, if we limit the evaluated angle range to $\pm \vartheta_{max}$,

$$T_W \gg \frac{L}{c} \sin \vartheta_{max} \tag{6}$$

If the actual windows length is

$$T_W = \varepsilon \frac{L}{c} \sin \vartheta_{max} \quad , \quad \varepsilon \geq 1 \tag{7}$$

then for the loss in directional gain D_{LOSS} in the reception of noise signals, we can calculate

$$D_{Loss} = 10 \log \left(1 - \frac{1}{3\varepsilon}\right) \tag{8}$$

This loss is 1.8 dB for $\varepsilon = 1$ and drops rapidly as ε increases.

For deterministic signals there is a loss of up to 6 dB, but this can be completely avoided by increasing ε and in processing of overlapping time windows.

3. Realisation with an Array Processor

When realising the real-time beam-former with an array processor, attention must be paid to the nature of the signals to be evaluated. It is not possible to deal with all types of signals in the course of this lecture. In the following, therefore, we shall confine our remarks to the processing of broad-band noise signals.

3.1 Algorithm

Fig. 5 illustrates the structure of the FFT beam-former. The temporal and spatial FFT's are shown. Then the envelopes of the signals are calculated by means of rectification and low-pass filtering. The summation on the lines ϑ = const. follows.

Fig. 6 shows a sketch of the chosen algorithm. We shall mention only the important operations here (for a more detailed treatment, see [4]).

N input signals are sampled and subjected to temporal FFT. For beam-forming, a band of interest with band width

$$B_{BF} < \frac{f_s}{2} \tag{9}$$

is selected from this and is reordered in the spatial direction. If required, amplitude shading can be carried out at the same time. The extra computer time is of no great importance.

Zeros are used for filling in to give interpolated values, and the spatial energy spectrum is calculated. Low-pass filtering follows for every point.

While computation is taking place, the data for the next computation cycle are read in per DMA (direct memory access).

The reading in of data is briefly interrupted after a freely selected number of such computation cycles, in order to make a single calculation of the broad-band directional pattern. This is transferred to the host computer, which manages the output devices. There should be an output every one or two seconds.

3.2 Program Sequence

Fig. 7 shows a diagram of the program time-squence. It can be seen that the main work - data intake, computation and filtering - is done in the array processor, and that the host computer just attends to the output of data. All activities take place simultaneously; data intake is interrupted briefly just for the calculation of signal power.

Parameters can be input to match the program to the external conditions. They determine internally

- o the antenna length
- o the frequency band evaluated
- o the time between outputs
- o the choice of output devices
- o the nature of the output.

Standard output devices are the refreshing display and the grey scale recorder.

3.3 Memory Capacity Requirement

A prerequisite for real-time beam-forming is simultaneous computation and data intake. Memory organization is therefore as follows:

- o Input buffer
- o Working memory
- o Result accumulator

The size of the input buffer is fixed by

$$N_{IN} = f_s \cdot T_w \cdot N \tag{10}$$

The necessary sorting operations and the faster "not in place" FFT require a working memory of double the size.

$$N_{WO} = 2 \cdot N_{IN} \tag{11}$$

The size of the result accumulator is determined by the bandwidth evaluated and the degree of interpolation in the spatial FFT.

$$(12) \qquad N_{AC} = B_{BF} \cdot T_w \cdot N$$

A small storage region is also required for constants.

The memory requirement is determined mainly by the product of the sampling frequency and the window length. For

$$f_s \cdot T_w = 256,$$

beam-forming with up to 16 channels can be performed with a 16 K data memory.

For 40 channels, more than 32 K is required.

3.4 Speed of Operation

The computation time T_c also depends mainly on the product

$$f_s \cdot T_w$$

according to a function

$$(13) \qquad T_c = \alpha \cdot N \cdot f_s \cdot T_w + \beta \cdot B_{BF} \cdot T_w$$

where α, β are hardware constants.

If the computation times quoted for the AP 120 B are summed, we obtain

29.3 ms

for 8 channels and $f_s \cdot T_w = 256$. For every additional 8 channels there are another 9 ms. A little more time is needed for some minor management tasks. The computation time measured for 8 channels was 30 ms.

For gapless reading in with 8 channels there is thus a maximum sampling frequency of

8.5 kHz per channel

and with 40 channels it is still

3.9 kHz per channel.

The single calculation of the broad-band directional pattern requires another 17 ms.

4. Modifications

A large variety of modifications can be effected by means of simple program changes. Thus, for example, output can take place with linear or logarithmic scales and with any normalisation. The extra time required for conversions to dB, for example, is only 1.5 ms per output and is thus of no great significance.

The use of time windows or frequency weighting can also be implemented with little trouble. The Hanning window can be quoted here as an example. With a very slight programming effort the additional execution time for 256 points is about 200 μs.

However, a reformatting and reordering operation is required in any case. With a little more programming effort, therefore, the Hanning window can be implemented with no extra time requirement.

Conclusions

Beamforming is possible by means of the combined temporal and spatial FFT. This type of signal processing is ideally suited for an array processor. The system described in this paper was already successfully tested on board a research vessel. The current system is able to process up to 16 individual input channels. The maximum band-width is fixed to 1 KHz. It is planned to increase the memory capacity so that the processing of up to 40 channels is possible in real time.

REFERENCES

[1] Roger G. Pridham, Ronald A. Mucci:
A noval approach to digital beam-forming.
JASA Vol. 63 No. 2, 1978, pp. 425, 434.

[2] Jack R. Williams:
Fast beam-forming algorithm.
JASA Vol. 44 No. 5, 1968, pp. 1454 - 1455.

[3] R. Diehl:
Richtungsbildung in Echtzeit mit einem Array Processor (real-time beam-forming with an array processor).
BL 5537, 1978, Krupp Atlas-Elektronik.

[4] Walter G. Wagner:
Realisierung eines Beamformers mit einem Array Processor (realising a beam-former with an array processor).
BL 4609, 1979, confidential, Krupp Atlas-Elektronik.

DISCUSSION

C. Richardson You describe computation of the broadband directional noise spectrum within the AP. However, the linear array may be subject to low velocity coherent energy, i.e., $k > \omega/c$ [k = wave number or spatial frequency], which is, therefore "outside" the directional arc. Have you investigated this? (The question relates to array design rather than operational performance.)

W.G. Wagner What we are dealing with in this paper is the implementation of a beamforming algorithm with an array processor. In the introduction I gave the boundary conditions belonging to the physical situation.

J.M. Griffin You stated that the memory requirements for the work area is twice the channel-input sample size product. Since only one channel is transformed at a time why was so much memory allocated?

W.G. Wagner The input data is in channel order first and then in time order, so a re-ordering algorithm is required. For simplicity, and in order to reduce programming effort, more memory space than necessary was used. As it is intended to expand the number of input channels, the program structure had to be modular so no extra programming was required.

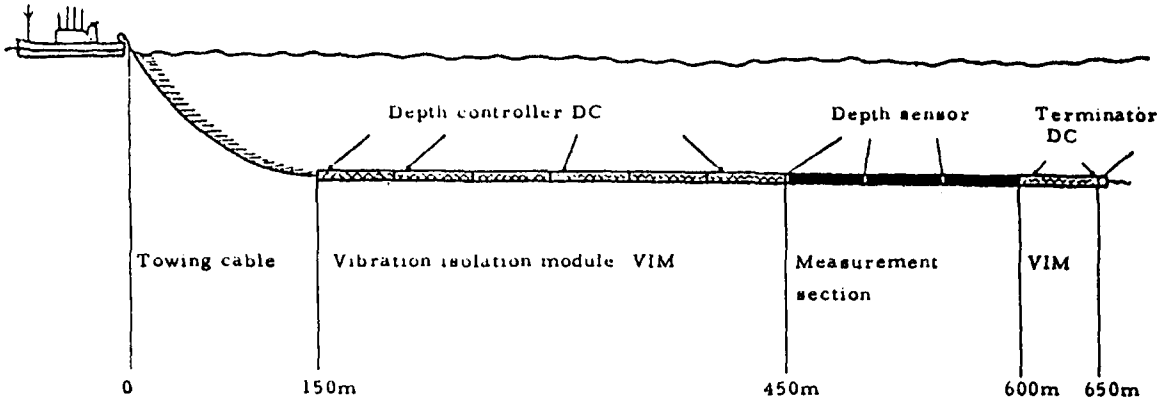


FIG. 1 TOWED ARRAY SYSTEM OVERVIEW

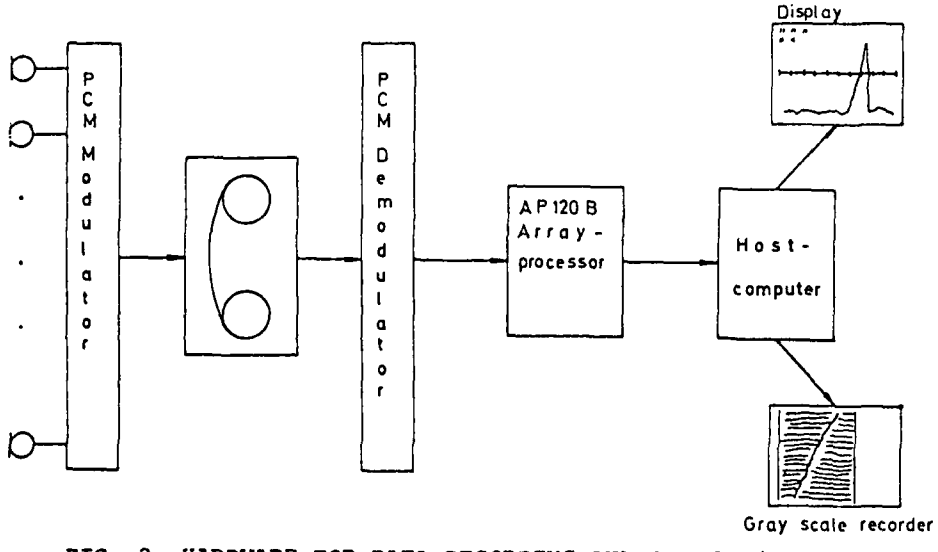


FIG. 2 HARDWARE FOR DATA RECORDING AND ON-LINE/OFF-LINE EVALUATION

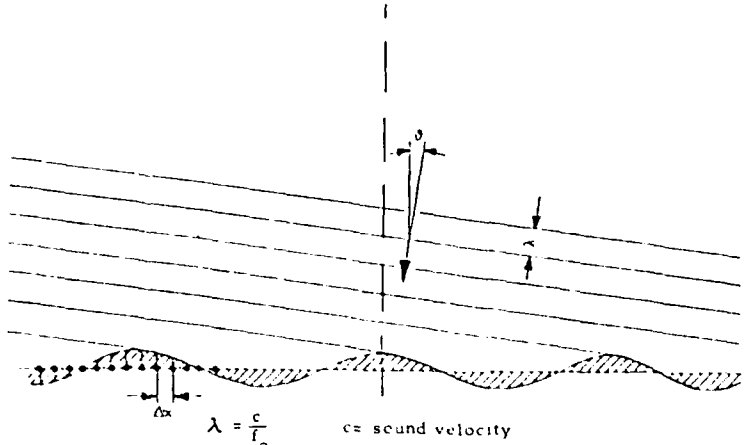


FIG. 3 NOTATIONS

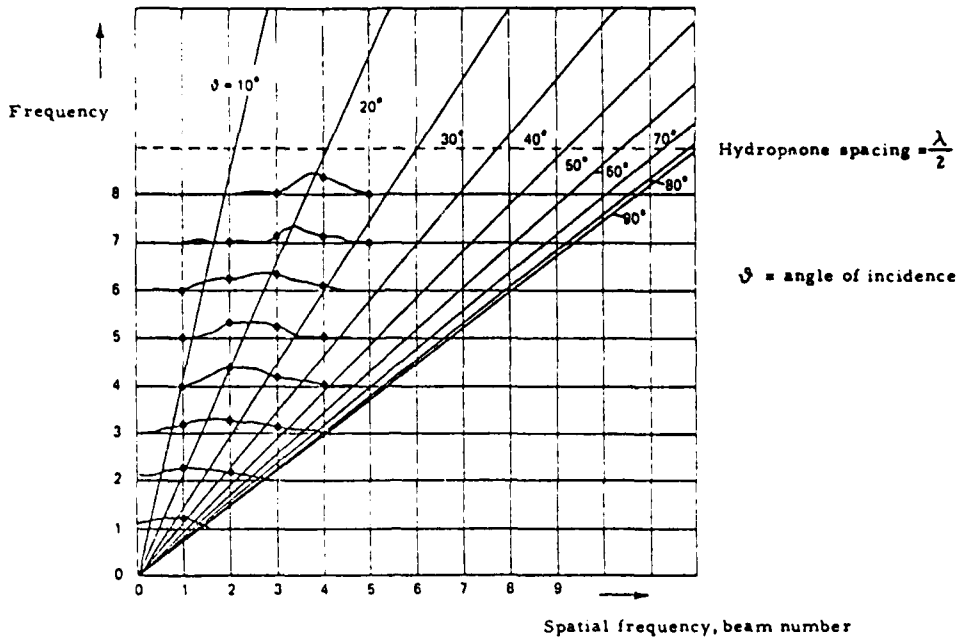


FIG. 4 RELATIONSHIP BETWEEN SPATIAL FREQUENCY AND SIGNAL FREQUENCY

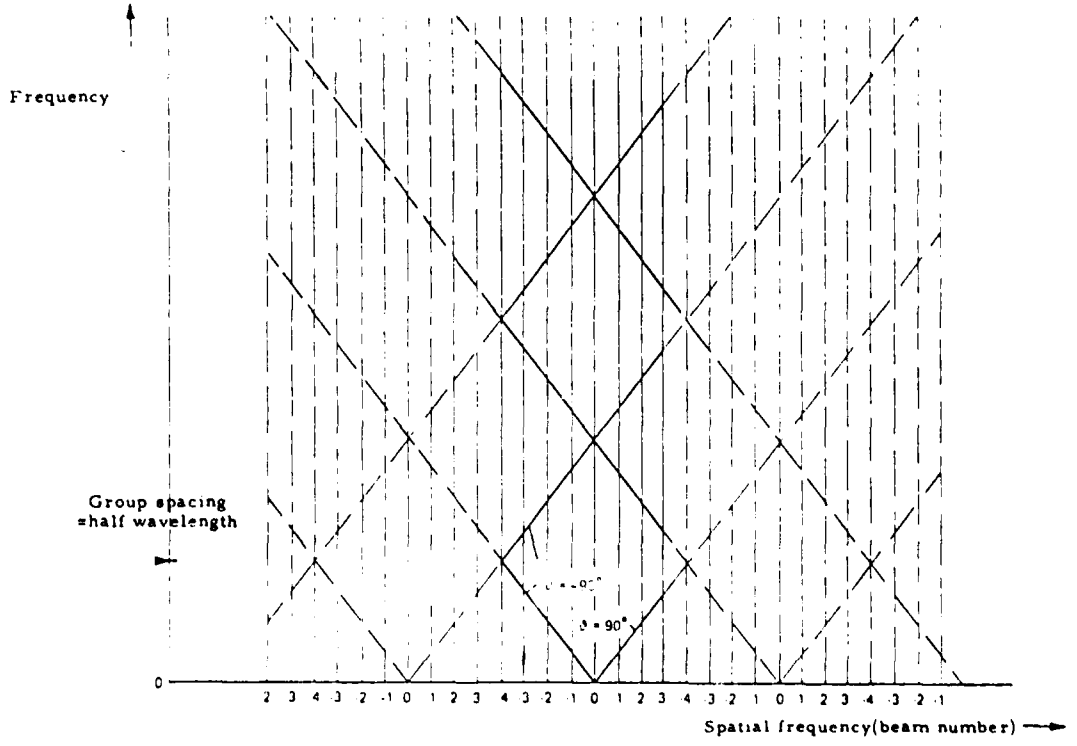


FIG. 5 SPATIAL PERIODICITY WITH FFT BEAMFORMING

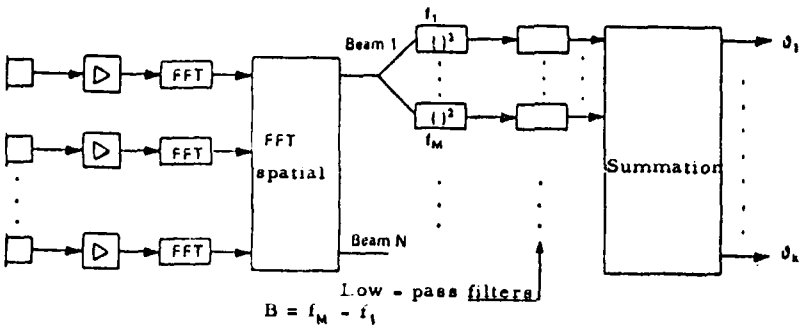


FIG. 6 FFT BEAMFORMING - STRUCTURE

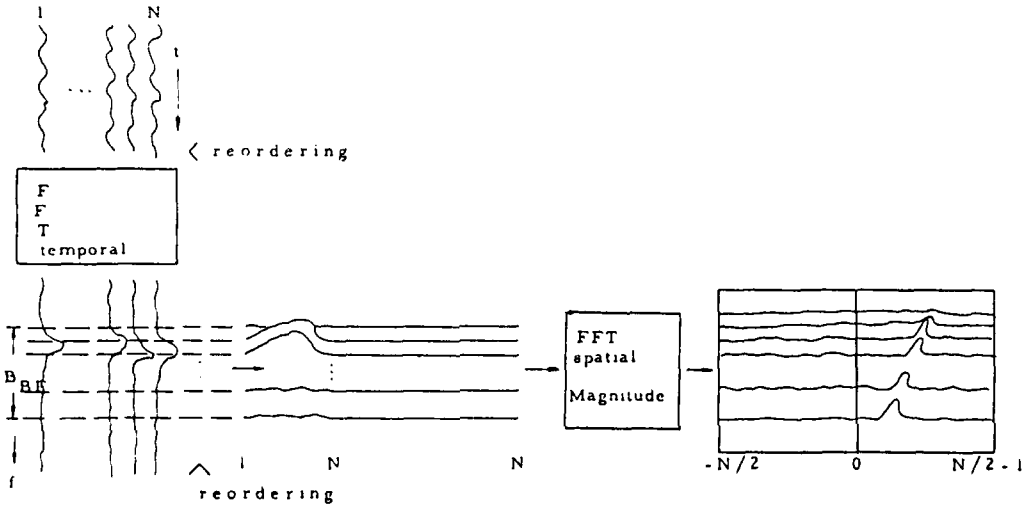


FIG. 7 SKETCH OF THE ALGORITHM

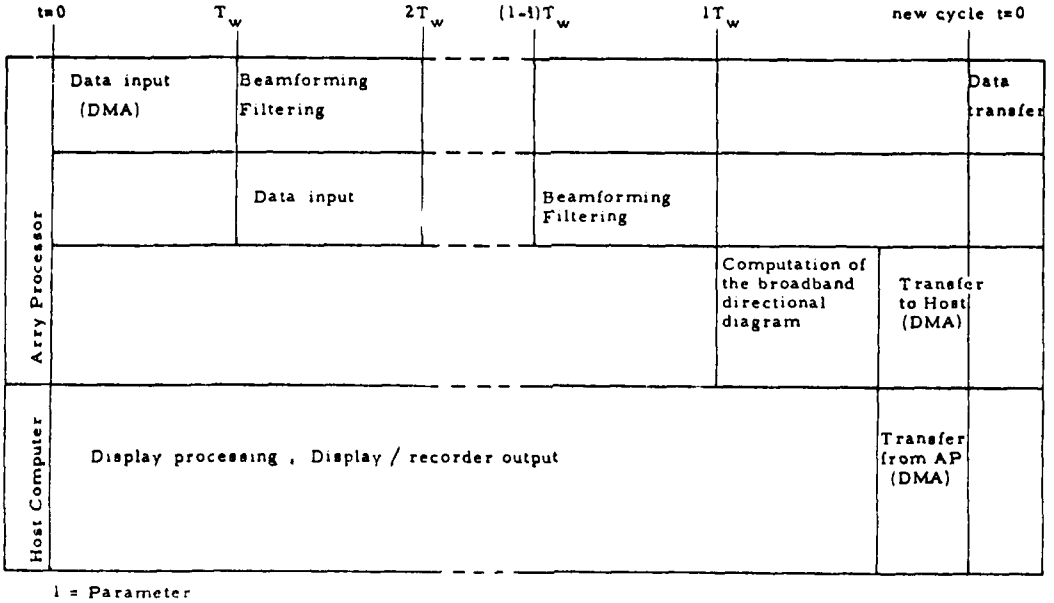


FIG. 8 TIME SEQUENCE OF BROADBAND SIGNAL PROCESSING

AN INTRODUCTION TO ADAPTIVE ARRAY PROCESSING

by

J.W.R. Griffiths
Department of Electronic and Electrical Engineering
University of Technology, Loughborough, Leics.,U.K.

1. INTRODUCTION

Array processing systems which can respond to an unknown interference environment are currently of considerable interest. The fundamentals of such systems are by no means new - basically they depend on Wiener's filter theory - but application in practice has been limited both by technology and by the lack of robust algorithms rapid enough for real time operation. Rapid strides in the past decade or so in the twin fields of electronic components and computer technology have changed the situation considerably by offering the possibility of complicated signal processing in real time at economic costs. This has led to an increased interest in adaptive processing and although it is probably still finding its main application in the defence field the civilian applications are growing. As a consequence the literature on this subject has expanded rapidly but as this paper is intended for presentation rather than wide publication the number of references has been kept to a minimum. Ample references - and indeed good papers - can be found in the special issue of the IEEE Transactions[1] devoted to this subject.

An array comprises a set of sensors, the outputs of which are combined in some way to produce a desired effect, e.g. a set of beams 'looking' in various directions. The sensors may be of many forms, e.g. acoustic transducers for sonar, monopole aerials for h.f. reception, microwave horns in a radar system, the influence of the application on the processing required concerns the technology rather than the principles involved. Sensors may be distributed in space in various ways but the two most common are the linear array, normally a set of equally spaced sensors in a straight line and the circular array in which the sensors are arranged uniformly in a circular pattern. Although the distribution of sensors obviously affects the problem the effect in general is only of second order. The linear array, apart from being the most common in practice, is also the simplest to describe and understand and will be used in this paper as the basic model. The introduction to the theory will also be limited to 'band pass' systems, i.e. systems in which the signals can be described in terms of a carrier with a complex amplitude. In the theory the carrier will thus not be explicit and the calculations will be based on the complex numbers representing the amplitude and phase at any time.

In general wide-band systems can be dealt with by using an F.F.T. processor to provide a set of narrow-band systems to which the adaptive method is applied individually.

2. BASIC SYSTEM

The system which we are considering is shown in Fig. 1 and comprises a linear array of K equally spaced elements, the outputs of which are individually weighted before being summed. Basically we wish to adjust the weights in an adaptive manner such that the ratio of the output due to the signal (or noise) being received from a given direction compared to that due to the signals (or noise) arriving from other directions is optimised. We will define the output as

$$y = \sum_{i=1}^{K} w_i x_i \qquad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

where x_i are the individual element voltages and w_i the weights. The values of x_i will be real for a simple low pass system but for the more common bandpass systems the values will be complex since they represent the amplitude and phase of the received signals. In general the weights will be complex and hence also the output y. In practice we may be dealing with continuous or sampled values of the variables but since in the first part of the analysis we shall be using averages or expectation of the variable the method applies to either sampled or continuous systems.

For simplicity it is convenient to use vector notation and so we define the output as

$$y = W^{\dagger} X = X^{T} W^{*} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

where

$$W \triangleq \begin{vmatrix} w_1^* \\ w_2^* \\ \vdots \\ w_K^* \end{vmatrix} \quad \text{and} \quad X \triangleq \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{vmatrix}$$

T indicates the transpose
† indicates the complex transpose
* is a conjugate without transpose

The mean output power is given by

$$\begin{aligned} P &= E\left[|y|^2\right] = E\left[y.y^*\right] \\ &= E\left[W^{\dagger} X \,.\, X^{\dagger} W\right] \\ &= W^{\dagger} R_{xx} W \end{aligned} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

where $R_{xx} \triangleq E\left[X \,.\, X^{\dagger}\right]$

R_{xx} is known as the Correlation Matrix. If the variables x_i have zero mean then R_{xx} can also be referred to as the Covariance Matrix.

3. STEERING VECTOR CONSTRAINT

If we know what is the desired output waveform y_D we may minimise the mean square error between y and y_D and in some systems this is possible. We shall be considering situations in which we do not know y_D and hence we need to constrain the system in some manner. One simple method is to constrain the gain in the wanted direction. This can be achieved by defining a steering vector C which is basically a vector representing the signals which would be present on the elements if a plane wave was arriving from the wanted direction θ viz:-

$$C^{\dagger} \triangleq \left[1,\ e^{-j\phi},\ e^{-j2\phi} \ \ldots\ldots\ e^{-j(K-1)\phi}\right] \qquad \ldots\ldots\ldots\ldots\ (4)$$

where $\phi = \frac{2\pi d}{\lambda} \sin\theta$

d = spacing between elements

λ = wavelength of plane wave signal

We wish to constrain the system such that the output from this signal is unity thus

$$W^{\dagger} C = 1 \qquad \ldots\ldots\ldots\ldots\ (5)$$

Since this constraint maintains the gain to the wanted signal constant we can now minimise the output subject to this constraint. To do this we use the method of Lagrange Multipliers and generate a cost function.

$$H(W) = P + \lambda\,(1 - W^{\dagger} \cdot C) \qquad \ldots\ldots\ldots\ldots\ (6)$$

Differentiating w.r.t. to W gives

$$\nabla H(W) = 2\,R_{xx}.W - \lambda.C \qquad \ldots\ldots\ldots\ldots\ (7)$$

Equating to zero and using the constraint to determine λ we get an expression for the optimal weight vector W_o.

i.e. $$W_o = \frac{R_{xx}^{-1} \cdot C}{C^{\dagger}.R_{xx}^{-1}.C} \qquad \ldots\ldots\ldots\ldots\ (8)$$

Substitution of this optimal solution in the equation for the output power gives up the value for the optimal output power P_o.

i.e. $$P_o = \frac{1}{C^\dagger R_{xx}^{-1} C} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (9)$$

Hence $$W_o = P_o \cdot R_{xx}^{-1} \cdot C \qquad \ldots\ldots\ldots\ldots\ldots\ldots (10)$$

Fig. 2 shows the operation of a two element system in which the weights and inputs are assumed to be real. Although this is a gross over-simplification it does, however, illustrate a number of aspects of the behaviour of the more complicated systems. As will be seen from the diagram the constraint equation $W^\dagger C = 1$ defines a line (a surface in the multi-dimensional case) and the tip of the weight vector must lie on this line in order to maintain the gain as unity to the steering vector. The conventional weight vector lies along the direction of the steering vector as is shown. An interference vector can represent an unwanted signal arriving from a particular direction and the optimal solution is when the weight vector is in a direction orthogonal to that of the interference vector, i.e. when the output due to the inner product of these two vectors is zero. An illustrative example is shown in the figure.

This diagram can also be used to illustrate a problem that results from this simple method of constraint. It can be seen that unless the wanted signal is arriving from exactly the direction of the steering vector then in theory it will be nulled out. In fact, of course, uncorrelated noise is always present and as the length of the weighting vector increases the contribution from such noise in the output will increase. This then puts an effective limit on the magnitude of the weight vector. Fig. 3 shows the effective beam pattern of a simple constraint system for various signal/uncorrelated noise ratios. It can be seen that when the signal/background noise is high the beam is very narrow compared to the conventional beam formed by equal weighting of the element outputs.

4. THE REAL TIME PROBLEM

It can be shown that the solution derived in the last section is analogous to the matched filter approach for detecting a pulse in coloured noise. Thus if we are dealing with a stationary system and are not in a hurry the solution is fairly straightforward. The strategy would be to steer the system in as many directions as required and calculate the optimum 'weight vector' for each direction. This would enable the power received from each direction to be measured under the optimal conditions. Normally in practice neither the condition of stationarity, nor of ample time, hold and what we require is a system which can adapt to changing conditions and preferably do this very rapidly.

The most obvious approach is to use a recursive method and to update the weights as new data is processed. We have seen that our criterion is to minimise the expected mean square of the output (l.m.s.) subject to the constraint. If we imagine a multi-dimensional space whose ordinates are

the weight vector components then for a given environmental situation, i.e. a particular group of incoming signal interference/noise, a set of contours can be drawn for a given power output. These will form a 'bowl' and we are trying to seek the minimum (bottom) of this bowl subject to the constraint requirement. Fig. 4 illustrates this point using the very simple situation of a weight vector comprising two real components. One of the standard methods of searching for the minimum is to move along the direction of steepest descent. We require, however, also to satisfy our constraint and so we approach the solution in a series of steps.

Step I. We calculate the gradient vector from Equation (3) viz:-

$$\Delta = 2R_{xx} W \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad (11)$$

Step II. Using a projection operator we decompose this vector into two components, one of which is parallel to the constraint vector and the other is orthogonal.

Step III. We subtract from our present weight vector W_k a proportion β of the component of the gradient vector which is orthogonal to the constraint vector and hence the new weight vector W_{k+1} will maintain the required constraint condition.

$$W_{k+1} = W_k - \beta\{I - \frac{cc^\dagger}{c^\dagger c}\} \cdot R_{xx} W_k \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad (12)$$

Fig. 5 illustrates this process for the simple two weight example.

Unfortunately in a practical situation we do not know R_{xx} since this is the expectation of $XX^\dagger$. However, we can make an estimate from the data available, i.e.

$$\hat{R}_{xx}(k) = \frac{1}{k} \sum_{i=1}^{k} X_i X_i^\dagger \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad (13)$$

and use this in equation (12).

What appears to be a more drastic approximation but in fact has advantages is to use the present value $X_k X_k^\dagger$ as an estimate for R_{xx}. Substitution in Equation (12) gives

$$W_{k+1} = W_k - \beta \{I - \frac{cc^\dagger}{c^\dagger c}\} X_k X_k^\dagger W_k$$

but $X_k^\dagger W_k$ is the current output y_k

$$\therefore \quad W_{k+1} = W_k - \beta \{I - \frac{cc^\dagger}{c^\dagger c}\} X_k y_k$$

This method is known as the stochastic steepest descent algorithm and results in fairly simple hardware for implementation[2], an example of which is given in Fig. 6.

5. SAMPLE MATRIX INVERSION (S.M.I.)

The sample covariance matrix is defined for zero-mean data as

$$\hat{R}_{xx} = \frac{1}{M} \sum_{i=1}^{M} X_i X_i^{\dagger}$$

where M is the number data samples in the observation interval. The block may be formed in several ways, M can be equal to the total number of data samples available, or the data can be divided into blocks of M samples, or the sample matrix can be generated by a sliding average of M samples. The value of SMI in adaptive beamforming in ill-conditioned interference environments was discussed by Reed[3] et al who proposed the estimate for the optimum weight vector as

$$\hat{W}_o = \frac{\hat{R}_{xx}^{-1} C}{C^{\dagger} R_{xx}^{-1} C}$$

This is, of course, the same result as was obtained for the constrained solution discussed earlier. To determine R_{xx} requires something of the order of K^3 operations but the computational disadvantage of having to determine R_{xx}^{-1} may be compensated by the fact that convergence is normally faster for ill- conditioned environments. Another advantage is that if the problem is such that we are required to determine the angular distribution of the power received rather than concentrate on the signal from one direction then once R_{xx}^{-1} has been calculated the computation to obtain the appropriate optimum vector for each direction is relatively trivial. It should be pointed out that if the data is dealt with in blocks some method has to be used to prevent signal suppression, e.g. addition of white noise or limitation on the weight vector norm, as discussed earlier. A comparison of the SMI technique with the stochastic steepest descent is illustrated for a three-jammer environment in Fig. 7.

6. CONCLUSION

This paper has dealt with a relatively limited problem and only some of the possible solutions. We have not discussed the effect of truncation and quantization in the processing nor the need for robustness in dealing with, for example, variations of the sensitivities or positions. Suffice it to say that these introduce further complications but are capable of analysis and control. There is no panacea since the 'best' solution varies

according to the environment in which it is to be applied. In simulation it is fairly easy to set up data for which a particular algorithm works well but just as easy to produce data for which it does not!

7. ACKNOWLEDGEMENT

The author would like to acknowledge the many sources from which information has been obtained and in particular the assistance of his colleague, Dr. J.E. Hudson.

8. REFERENCES

(1) Special Issue on Adaptive Arrays, IEEE Trans. Ant. and Prop. 1963.

(2) Frost, O.L., "An algorithm for linearly constrained adaptive array processing", Proc. IEEE, Dec. 1967, No. 12, pp 2143-2159.

(3) Reed, I.S., Mallett, J.D., Brennan, L.E. "Rapid convergence rate in adaptive arrays", IEEE Trans. Aerospace AES10 (1974) pp. 853-863.

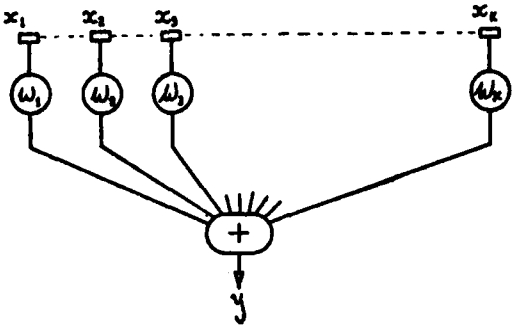
DISCUSSION

R. Seynaeve Can you comment on the accuracy required by the various algorithms?

J.W.R. Griffiths Depends on the problem. With the constraint technique, sensor inaccuracies in both position and gain can be tolerated. The number of bits depends on the noise "floor level".

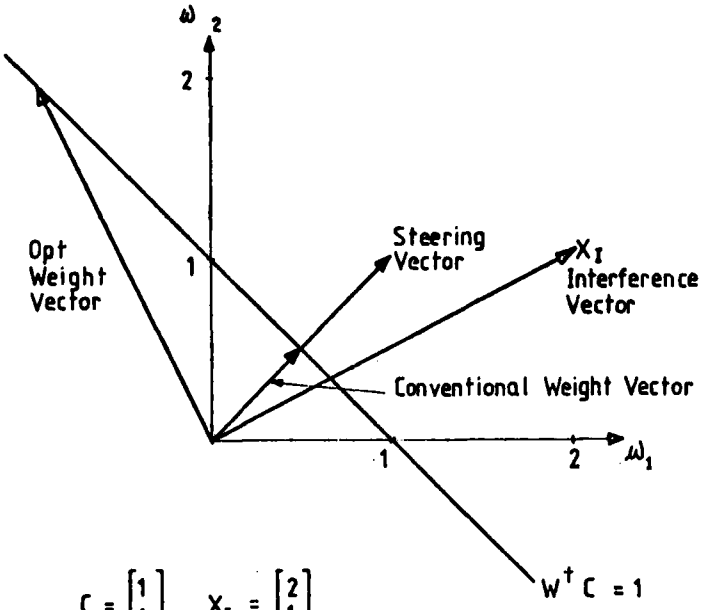
H.J. Alker In the U.K. are the real-time adaptive systems using steepest-descent or matrix-inversion techniques in the adaptive processing?

J.W.R. Griffiths Both are being considered.



Basic System.

FIG. 1



$$C = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad X_I = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$W_o = \begin{bmatrix} -1 \\ 2 \end{bmatrix} \qquad W_o^T C = \begin{bmatrix} -1, 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 1$$

$$W_o^T X_I = \begin{bmatrix} -1, 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = 0$$

Two element system with steering vector constraint.

FIG. 2

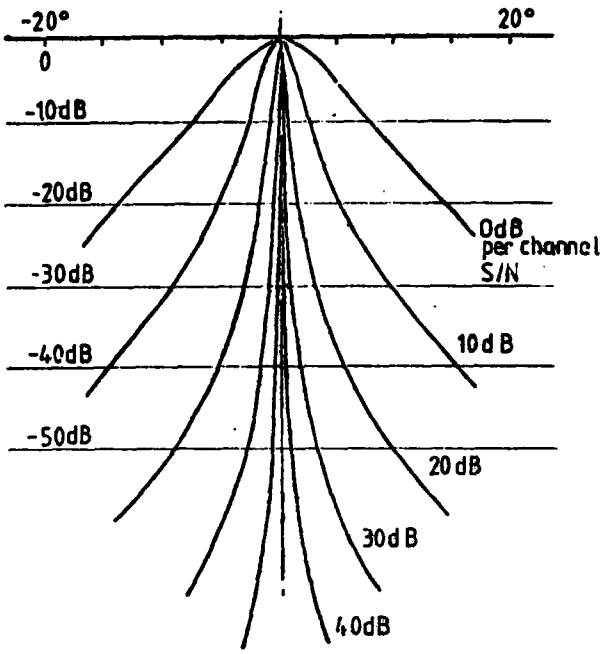


FIG. 3 Signal output power against angle. zero moment constraint, 0dB signal.

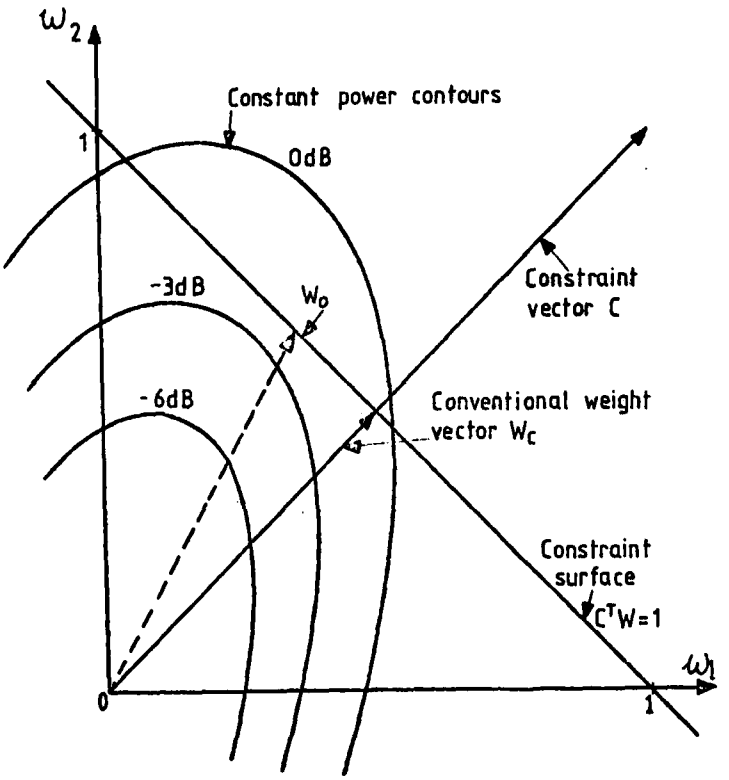
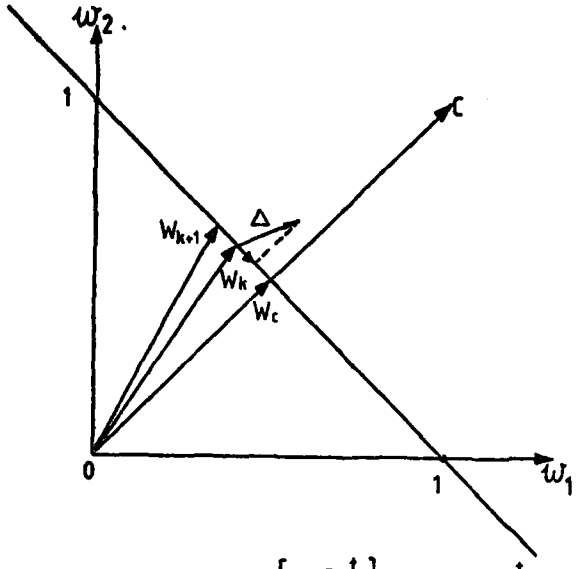


FIG. 4 Simple case showing equal power contours.



$$W_{k+1} = W_k - \beta\left[1 - \frac{C C^\dagger}{C^\dagger C}\right] R_{xx} W_k$$

FIG. 5

Steepest descent algorithm .

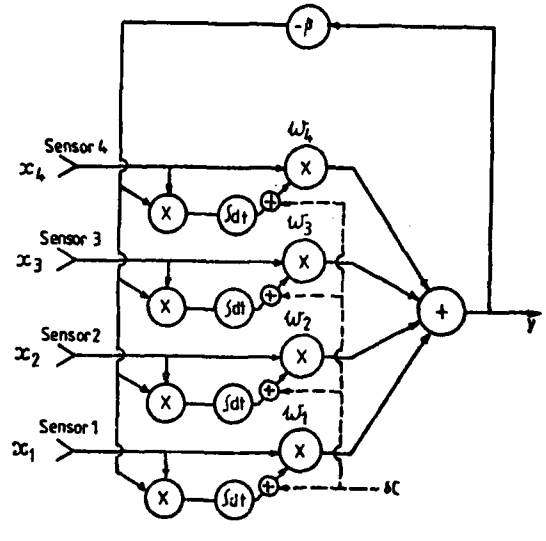
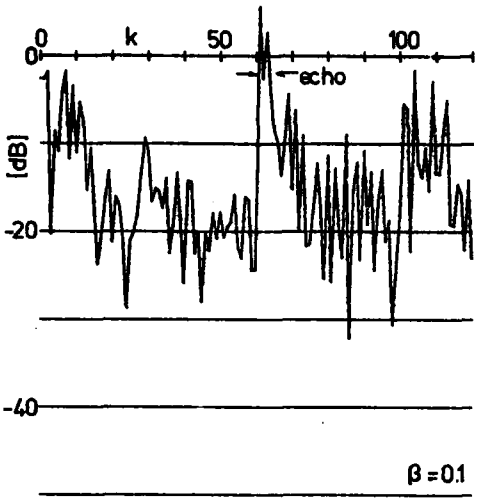
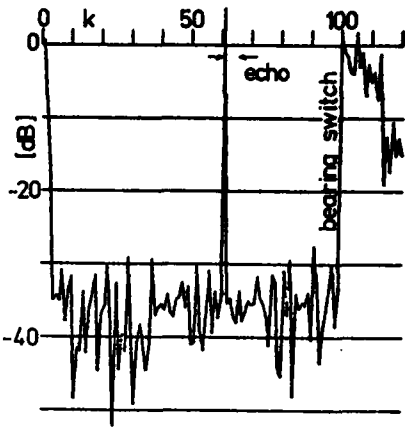


FIG. 6

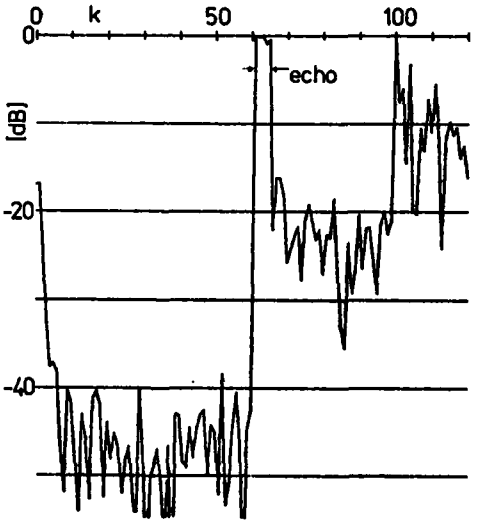
A Narrow-Band Array-Space Maximum Likelihood Circuit (After Frost.)



3 JAMMERS SIDELOBE CANCELLER
STOCHASTIC STEEPEST DESCENT.



3 JAMMERS: 0,-10,-20 dB NOISE:-40dB
N_e=5 SIDELOBE CANCELLER
SAMPLE MATRIX INVERSION



3 JAMMERS: 0,-10,-20 dB at -45°,-20°,+45°
N_e =5 NOISE 45 dB BROADSIDE CONSTR'NT.
SAMPLE MATRIX INVERSION.

FIG. 7

PANEL ON BEAMFORMING

Chairman: T.E. Curtis

Members : A. Baggeroer
D.V. Crowe
J.W.R. Griffiths
H. Urban
W.G. Wagner

T.E. Curtis The systems described so far are O.K. for linear arrays. Has anyone done any work on other arrays using FFT-based processing?

D.A. Crowe Using sparse matrix beamforming, we found that if we lost a hydrophone in our equi-spaced array, we often obtained spurious results; however, errors were predictable if coherence over the array was analysed. I am not sure that time-domain beamforming would always help.

T.E. Curtis True, time domain could still have problems, unless the system is well monitored. There is also the problem of low-frequency acceleration of the array along beam directions, and these must be compensated.

A. Baggeroer We used an L-shaped array. It was easy to compensate for a lost hydrophone.

J.W.R. Griffiths I don't think FFT is a good technique in a non-stationary environment.

H. Urban Beamforming with array processors is good for experimental work. However, I think hard-wired beamformers are better suited to operational systems.

J.M. Griffin Has anyone an experience with maximum entropy beamforming?

A. Baggeroer The problem of non-equispaced arrays is still unsolved, except by brute force. Prediction error algorithms are used extensively in maximum entropy techniques - they are available off the shelf - TI make a chip. TI's "speak and spell" device uses a lattice type network; the theory is connected with prediction algorithms.

T.E. Curtis I think I'm the only hardware man at this Conference, and I would like to change emphasis. Can anyone suggest where array processors are going? In the U.K. we have examined available machines such as Cray 1's and find that the FFT approach even on a large machine still does not give what we need.

R. Seynaeve CCD technology looked set for a breakthrough in beamforming for several years, but is no longer in the news - please comment.

T.E. Curtis CCD's have had a bad press. The trouble was that people were trying to use CCD as a replacement for conventional technology - things such as tapped delay lines, which are hard to implement on a chip. The basic architecture was wrong - designers should concentrate on what CCD can do well, such as serial to parallel conversion. This is starting to happen in some interim areas (both analogue and digital); however, the PCM people have switched interest from CCD to switched capacitor circuits which has taken some interest away from analogue CCD.

R. Seynaeve So you see nothing new there in the next three years.

T.E. Curtis No, except in special purpose applications - for instance, low cost beamformers for fish-finding sonars, in low-power systems restricted in physical size, and in CCD logic.

H. Urban CCD has been used successfully using available chips - you only need a technician, some data sheets and six months.

R. Seynaeve Why isn't it more popular?

H. Urban There are still some problems to be investigated - no high temperature capability, for instance.

R. Seynaeve So there is no flexible CCD beamformer.

H. Urban This is hardware, so it is not flexible.

R. Seynaeve It could be designed to be flexible.

H. Urban Ours wasn't.

T.E. Curtis Analogue CCD problems are: limited temperature range, intermodulation, input linearity, clock interference, and dynamic range. These could be overcome, but digital systems are well developed and freely available.

R. Seynaeve So, a CCD system might be O.K. for monitoring, but not for accurate work.

G.C. Vettori What about buoy systems - an FFT in a buoy for instance?

T.E. Curtis Then you may have to use CCD, but why process in a buoy? It would be simpler to use a radio link.

G.B. Pettersen A micro-processor could be used.

G.C. Vettori On 1024 points in a reasonable time?

G.B. Pettersen Yes.

T.E. Curtis It would be quite possible if you could afford a few watts.

A. Baggeroer There is an interesting report on FFT's at a 5 Hz frame rate.

T.E. Curtis All sorts of things are available now, chirp-z transforms for instance, but there are still some problems in using devices like this. I have seen a card using these chips that had seven potentiometers to set-up - this is not practical and these devices must be made easier to use.

G.B. Pettersen CCD's have trouble with sampling distortion at low clock rates.

C. Richardson Has anyone looked at sparsely-filled array techniques?

G.C. Vettori This technique is used in radio astronomy - there was a paper on this at the NATO Advanced Study Institute held in Porto Venere in 1976.

(C. Richardson gave a short talk on his experiences with this technique emphasizing the saving in data recording.)

R. Seynaeve In a number of applications it is very cost-effective in hardware terms to use a separate beamformer followed with an array processor. Beamformers are becoming cheaper.

A. Baggeroer Analogic are bringing out an array processor for about $10,000 next year.

R. Seynaeve Yes, it may be best to use array processors for the beamforming itself, if they are in mass production.

D. Nairn But software isn't free.

R. Seynaeve It's cheap for beamforming. There is really not much in the beamforming algorithm. However, hardware beamformers should be cheaper too.

T.E. Curtis It depends which way one wants to go - hardware or software.

R. Seynaeve Will your modules be commercialized?

T.E. Curtis May be - one card costs us around St. 250.

D. Nairn How about stabilization of beams?

R. Seynaeve Not difficult with software.

G.C. Vettori I thought it was provable that in terms of power, the output of a beamformer is equivalent to the output of a multiplicative structure.

A. Baggeroer Multiplicative structure is more susceptible to noise - you get the equivalent of FM capture effects.

G.C. Vettori You can remove sensors without affecting directionality, which is affected in linear systems.

A. Baggeroer It is a question of how you model.

C. Richardson I agree. We are certainly aware of noise susceptibility - this is one of the basic system limitations.

HIGH-SPEED SIMULATION OF AN UNDERWATER ACOUSTIC FIELD USING AN ARRAY PROCESSOR

by

W.A. Kuperman, F.B. Jensen, M.G. Martinelli
SACLANT ASW Research Centre
La Spezia, Italy

ABSTRACT

A computer model PAREQ for simulating the acoustic field of sound propagating in a complicated ocean environment has been installed on a real-time computer system (HP MX21) which includes an array processor (MAP 300). The use of the array processor has resulted in a reduction in running time of a factor of one hundred over previous usage on a general-purpose computer (UNIVAC 1106). Consequently, parametric studies, which prior to this installation were impractical to perform, can now be made. The model can be used for at-sea interpretation of experimental results and for the simulation of data which would be received by various acoustic arrays and analyzed by this same real-time computer system. Hence, array configurations and signal processing schemes can, to some extent, be tested before the actual sea trials.

INTRODUCTION

Computer modelling in underwater acoustics play an important role in understanding the physics of sound propagation in the ocean and therefore also has a significant impact in the area of sonar technology. In this paper we describe the goals of SACLANTCEN's modelling effort and how a real-time computer system with an array processor has been used to extend our modelling capabilities.

1 BACKGROUND

Computer simulation of underwater acoustic propagation is a component of the research, development, and usage associated with sonar system technology. Figure 1 illustrates the role of modelling (computer simulation) by showing a schematic of a part of the sonar technology area. The schematic is divided into two horizontal levels. The upper level may be roughly

characterized as physics in which we are mainly concerned with understanding sound propagation in the ocean. This overall problem is approached both experimentally and theoretically (modelling). Experiments not only study sound propagation itself but also include simultaneous measurements of the ocean environment so that we may physically understand how sound propagates in the ocean. The modelling involves an attempt to describe propagation through this environment using the basic equations of physics. Hence the experiments are a check on the physical models, while at the same time models aid in the design of experiments.

For the second level of Fig. 1, there are three main aspects of development and usage:

1. Forecasting: predicting basic performance of specific sonars on a daily basis, based on the local ocean environmental conditions.

2. Analysis and performance prediction of existing systems: such studies can be used to optimize system performance by performing operations research type investigations.

3. Sonar system design: simulated data allows the systems designer to evaluate proposed systems under various environmental and operational conditions. Such studies require rather sophisticated modelling since future systems of, for example, extended aperture arrays require knowledge of the complex structure of the acoustic field.

Let us now briefly review the environmental acoustics of the ocean in order to make clear the complexity of the modelling problem. Figure 2 is a schematic of some possible propagation paths in the ocean. The ocean is bounded above by a rough wavy surface and below by an irregularly shaped bottom whose acoustic properties are generally quite complex. We show two possible locations of sources of sound on the left of the figure and we see that the sound is propagating to the right over a changing water depth, which adds increased complexity to the problem. The two dashed lines are sound-speed profiles (in electromagnetic theory they are the equivalent of indices of refraction) and we see that they vary with both depth and range. These profiles also have a statistical component caused by fluctuations in the ocean due to internal gravity waves.

Lines A, B, C, and D in Fig. 2 schematically represent four possible propagation paths. The shape of the paths are determined by the location of the source and the sound-speed structure over the extent of the propagation. Path A from the shallow source is a "surface-duct" propagation, because the sound-speed profile is such that the sound is trapped near the surface of the ocean. Complicating this path is the irregularity of both the ocean surface and the lower boundary of the surface duct, thereby allowing sound to escape and hence insonify other parts of the ocean. There is also the possibility that the ocean environment changes sufficiently that the duct disappears as illustrated in Fig. 2. Paths B, C, and D are from a deeper source. Ray B, leaving the source at a small angle from the horizontal, will tend to propagate in the "deep sound channel" without interacting with the boundaries (surface and bottom) of the ocean. This is usually a very stable path (propagation distances of thousands of

kilometres are possible) but which can be interrupted by the water becoming shallower (as shown) or by a change in the ocean climate.

At slightly steeper angles (C) we have "convergence zone" propagation, which is a spatially periodic phenomenon of zones of high intensity near the surface. Here the path interacts with the ocean surface but not with the bottom. Typically, the periodicity of the zones is of the order of 30 to 50 km and two or three successive zones are the most one can utilize with a sonar because of the combination of decaying intensity with range and the "blurring" of the zones of high intensity by irregularities in the ocean.

The third path (D) is the "bottom-bounce path"; its cycle period is shorter than that of the convergence zone and only distances of one, or at most two, bounces can be utilized because of the reflection loss at the bottom. The right-hand side of Fig. 2 depicts propagation on the continental shelf (shallow coastal waters) where a complicated bottom structure combined with variable sound-speed profiles result in rather complicated propagation conditions not always suited for a simplistic ray picture representation. Finally, all of the above discussion is complicated by the fact that the propagation is also highly dependent on the acoustic frequency.

The above is only a brief description of sound propagation in the ocean, intended to convey an impression of the complexity of the problem. It should now be easy to see that computer models to describe sound propagation in the ocean based on physical principles are likely to be large and complicated. In fact, at SACLANTCEN, we are to a large extent interested in the region of Fig. 2, where the ocean bottom has a significant slope. In this region the propagation is sufficiently complex that we have not attempted to make a schematic diagram of how sound would couple from the deep water to the shallow water. However, an example of propagation over a sloping bottom as simulated by an acoustic model will be shown later.

2 THE ACOUSTIC MODEL AND THE ARRAY PROCESSOR

There exist many acoustic models of propagation in the ocean. Some are special to certain types of environments, while others are more general, these latter constituting very sophisticated computer programs. We shall not discuss the various models here but confine ourselves to a specific one, the PAREQ model [1] which is a large, general-purpose model that is able to simulate all the phenomena discussed in Ch. 1.

Figure 3 is a schematic of the algorithm used in the PAREQ model. Essentially, this model produces the acoustic field as a function of range and depth by "marching" out in range step by step away from the source. The computational cycle shown to the right of Fig. 3 is for a single range step. Each range step requires two complex FFT's and two complex vector multiplications. The elements of the vector multipliers change as a function of range because of the varying ocean environment, but the range steps required are generally so small that many cycles are passed before the vectors need to be changed.

The cycle shown in Fig. 3 is particularly suited for an array processor, which can do both vector multiplications and FFT's very fast. Hence, we have dedicated the MAP 300 array processor [2] to do the simple computational cycle of Fig. 3, while the HP MX21 host computer changes the vectors occasionally when the sound-speed profile or bottom properties change.

Before implementing the model on this system it was resident on the SACLANTCEN's UNIVAC 1106 system. Not only was the reduction in running time an important motivation for using the array processor, but also, from the algorithm represented in Fig. 3, it is obvious that only very little change in the software structure of the model was required. For other models, even less sophisticated ones, the numerics are such that installation of these models would essentially require rewriting the software in order to benefit from the speed of the array processor.

In this particular case, a speed factor of the order of 100 is gained by going from a UNIVAC 1106 without an array processor to a HP MX21 computer with an array processor. This means that computer runs that took several hours on the UNIVAC system can now be reduced to minutes. Furthermore, because running times for this type of model go up as the square of the frequency, studies can be made that were not possible at all on the previous system.

3 SOME RESULTS

Here we will briefly present some output results from the PAREQ model. Figure 4 shows a deep-water convergence zone result (Fig. 2, Ray C). Plotted is propagation loss versus range for the specific source and receiver depths. Here we see the zones of high intensity appearing periodically with range. This run took around three hours on the UNIVAC 1106 and has now been reduced to only about two minutes. The second example, Fig. 5, shows a shallow-water example. The important thing to realize about this run is that in shallow water, as depicted in Fig. 2, there is a continuous interaction with the bottom, which leads to the necessity of a very precise environmental prescription and to very long running times. Again, running times have been reduced from hours to minutes. As a final example, we study propagation of sound up a sloping bottom into a shallower water region. Figure 6 shows a range/depth plot of propagation-loss contours for sound propagating up a bottom slope (indicated by the dark line). Here we see that at certain ranges part of the sound is radiated into the bottom and hence lost from the water column. Such a phenomenon has been verified experimentally. Again, we emphasize, this is an extremely complicated environment that cannot be treated by simpler models.

SUMMARY AND CONCLUSIONS

A model to simuate an underwater acoustic field has been installed in an array processor connected to an HP MX21 computer with a resulting increase

in speed of a factor of a hundred over its previous running time on a UNIVAC 1106. This will allow us to perform multi-frequency studies of complicated ocean environments, studies that could not have been done in a reasonable time on the UNIVAC system. In addition, since this model is now working on SACLANTCEN's real-time sea-going computer, the model can be used for at-sea analysis and interpretation of experimental data. For example, an existing, towed device to measure sound speed can be used to feed data into the model so that an in-situ acoustic prediction can be made and compared with the actual experimental data in real time. This procedure will allow the investigation at sea of anomalous experimental features that could not usually be studied until the experiment is over and the ship is far from the area of interest. Finally, the real-time computer system is being used for processing and beamforming of data from large moving arrays. The existence on this same system of a model for simulating data could aid in the design of the optimum configuration of these arrays and in selection of optimal signal-processing schemes.

REFERENCES

1. JENSEN, F.B. and KUPERMAN, W.A. Environmental acoustic modelling at SACLANTCEN, SACLANTCEN SR-34. SACLANT ASW Research Centre, La Spezia, Italy, 1979.

2. SEYNAEVE, R. et al. SACLANTCEN real-time signal processing system (WARP 1), Paper 3 of these Proceedings.

DISCUSSION

J.M. Griffin Did you run into any memory restrictions in implementing this model?

W. Kuperman No. The looping algorithm storage is done in the MAP and the resulting field at each range step is stored in the host computer.

S.G. Lemon Could the propagation model be used for optimum mode selection for a sonar system? (Mode such as towing depth optimization.)

W. Kuperman Yes, in the sense that it simulates the acoustic field and therefore allows the user to vary array geometry and see the results.

E. Hug What graphics package did you use for the contouring slide showing continental slope modes?

W. Kuperman That graphic package was on the UNIVAC.

R. Seynaeve We are completing the graphics package for the HP.

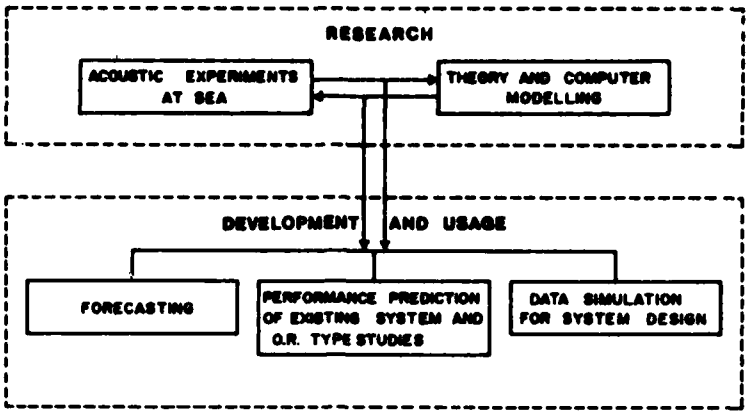


FIG. 1 ACOUSTIC MODELLING IN THE CONTEXT OF SONAR TECHNOLOGY

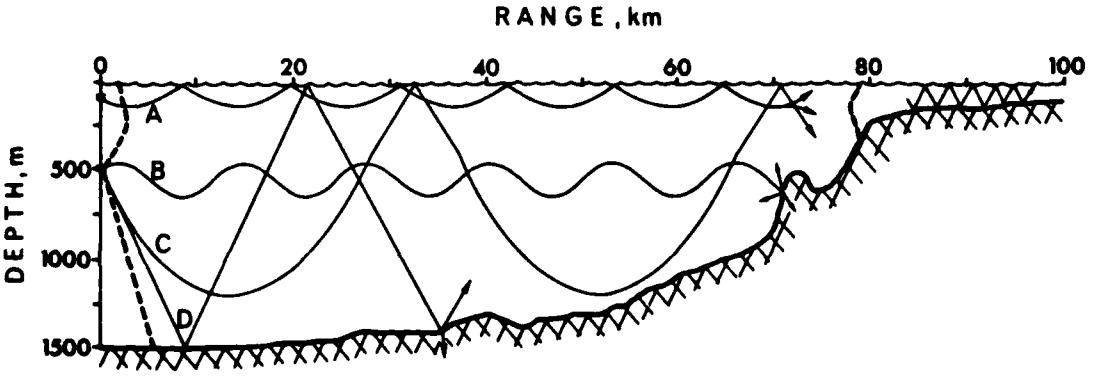


FIG. 2 SCHEMATIC REPRESENTATION OF SOUND PROPAGATION IN THE OCEAN

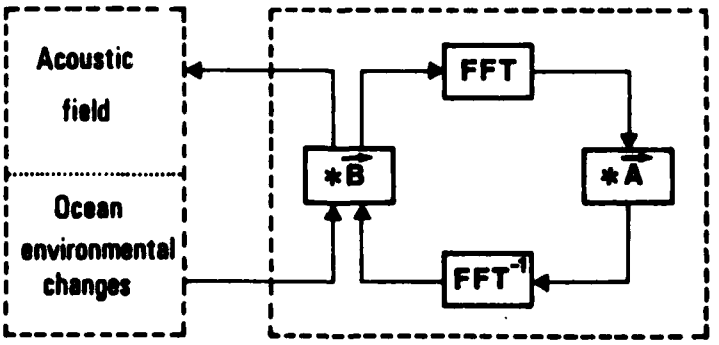


FIG. 3 MAIN COMPUTATIONAL CYCLE OF PAREQ MODEL

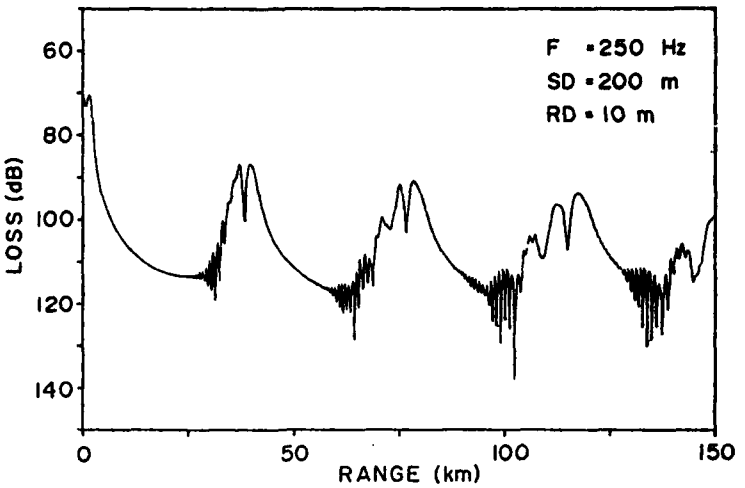


FIG. 4 PAREQ SOUND PROPAGATION PREDICTION FOR DEEP-WATER ENVIRONMENT

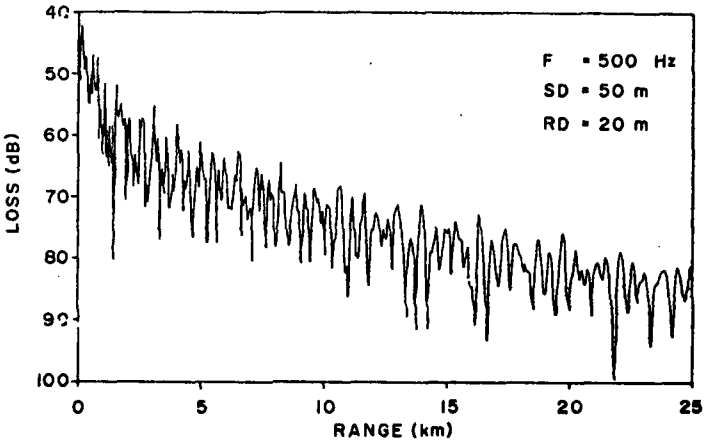


FIG. 5 PAREQ SOUND PROPAGATION PREDICTION FOR SHALLOW-WATER ENVIRONMENT

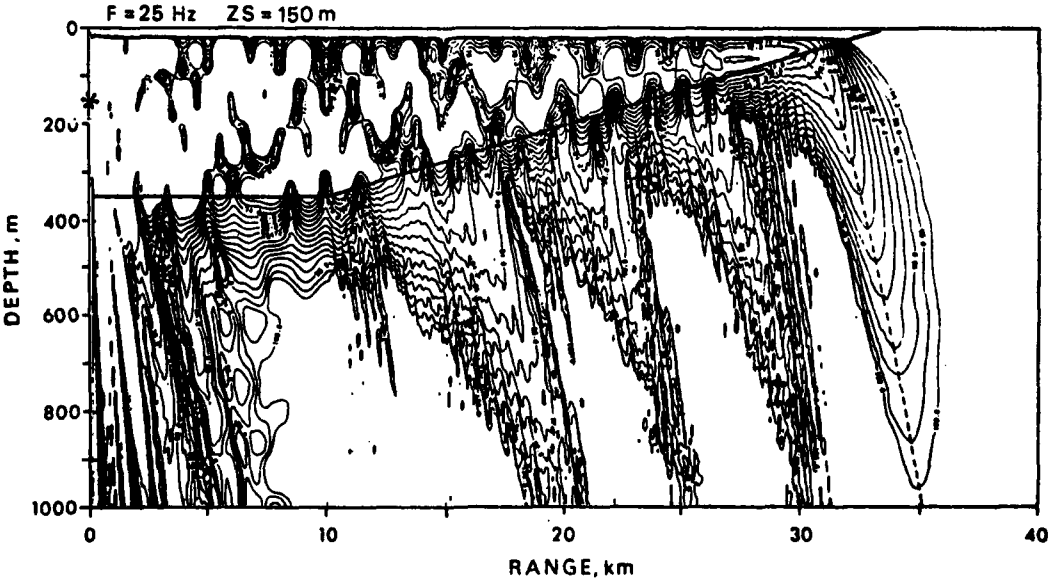


FIG. 6 CONTOURED SOUND FIELD FOR PROPAGATION OVER A SLOPING BOTTOM

SOME HIGH EFFICIENCY DIGITAL SIGNAL PROCESSING TECHNIQUES FOR HIGH-SPEED SIGNAL PROCESSING SYSTEMS

by

Vito Cappellini
Istituto di Elettronica, Facoltà di Ingegneria-Università
Florence, Italy

ABSTRACT Some high efficiency digital signal processing techniques, based on digital filtering, are presented for real-time general-purpose high-speed signal processing systems. At first high efficiency one-dimensional and two-dimensional digital filters are described, both of non-recursive FIR type (using Cappellini window) and of recursive IIR type (using transformations). Further a special technique, introduced by the author, for high-speed digital acquisition and processing with maximum efficiency, that is using the minimum number of sampled signal data for the required processing, is described. This technique using a single digital filter and a frequency shift procedure of the sampled signal spectrum, is particularly useful for performing a band-pass analysis or a spectral estimation with a tree-structure organization. Hardware and software implementations of the above technique are presented with typical application examples.

INTRODUCTION

Digital filtering is a very flexible and efficient technique for processing signals and images, in particular for underwater research. This technique can be useful in real-time general-purpose high-speed signal processing systems , both for reducing the amount of data and to process the important data according to any frequency characteristic which is required.

The reduction of the amount of data can be obtained through low-pass or band-pass digital filtering and subsequent "decimation" or, if the frequency behaviour of the signal is essentially required, through suitable spectral estimation algorithms.

In the following some high efficiency one-dimensional (1-D) and two-dimensional (2-D) digital filters are described. Hence a special technique is presented for high-speed digital acquisition and processing with maximum efficiency, that is using the minimum number of sampled data for the re-

quired processing. Hardware and software implementations are described with typical examples of results obtained in processing real signals having different frequency ranges.

1. SOME HIGH EFFICIENCY 1-D AND 2-D DIGITAL FILTERS

1-D digital filtering can be defined in general way by the following relation [1]

$$g(n) = \sum_{k=0}^{N-1} a(k)f(n-k) - \sum_{k=1}^{M-1} b(k)g(n-k) \qquad (1)$$

where: f(n), g(n) are, respectively, the input and output data (samples of the input and output signal); a(k) and b(k) are the coefficients defining the digital filter (its frequency response); N and M are two integers.

If all b(k) coefficients are equal to zero, the digital filter is called of finite impulse response (FIR) and the implementation structure is in general of non-recursive or trasversal type. If at least one b(k) coefficient is different from zero, the digital filter is called of infinite impulse response (IIR) and the implementation structure is in general of recursive type [1].

2-D digital filtering can be defined in analogous way by the following general relation

$$g(n_1,n_2) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} a(k_1,k_2)f(n_1-k_1,n_2-k_2) -$$

$$- \sum_{\substack{m_1=0 \\ m_1+m_2\neq 0}}^{M_1-1} \sum_{m_2=0}^{M_2-1} b(m_1,m_2)g(n_1-m_1,n_2-m_2) \qquad (2)$$

where: $f(n_1,n_2)$ and $g(n_1,n_2)$ are, respectively, the input and output data (samples of the original image and processed image); $a(k_1,k_2)$ and $b(m_1,m_2)$ are the coefficients defining the digital filter (its frequency response); N_1,N_2,M_1 and M_2 are suitable integer numbers [1].

Again, if all $b(m_1,m_2)$ coefficients are equal to zero, the 2-D digital filter is called of finite impulse response (FIR) and the implementation structure is in general of non-recursive or transversal type. If at least one $b(m_1,m_2)$ coefficient is different from zero, the 2-D digital filter is called of infinite impulse response (IIR) and the implementation structure is in general of recursive type.

Many methods have been proposed and defined to design and implement FIR

and IIR 1-D and 2-D digital filters [1] . Few methods are briefly recalled in the following, having good efficiency and relatively simple design and implementation properties.

For FIR digital filters (1-D and 2-D) a useful method is based on the use of suitable "window functions" to define the coefficients a(k) or $a(k_1,k_2)$. Indeed the problem of designing a digital filter in the frequency domain is connected to the evaluation of the coefficients in such a way that the obtained frequency response satisfies the required characteristics. In the window method we start from an impulse response, which is the inverse Fourier Transform of the desired filter frequency response: this impulse response, in sampled form h(k) or $h(k_1,k_2)$, is to be truncated for the practical implementation of the digital filter introducing the minimum error in the frequency response. To this purpose the values h(k) or $h(k_1,k_2)$ are multiplied by the samples w(k) or $w(k_1,k_2)$ of a suitable window function, having zero value in the region out of the truncation and a very high concentration in the frequency domain [1] . The obtained frequency response of the digital filter is therefore the convolution between $H(\omega)$ or $H(\omega_1,\omega_2)$, Discrete Fourier Transform (DFT) of h(k) or $h(k_1,k_2)$, and $W(\omega)$ or $W(\omega_1,\omega_2)$, DFT of w(k) or $w(k_1,k_2)$.

Many window functions are known for the design of FIR digital filters. If one window function is known for 1-D case, the 2-D extension is easily obtained, as shown by Huang [2] ,(with circular symmetry properties), by

$$w(x,y) = w\sqrt{(x^2 + y^2)} \tag{3}$$

Two window functions of particular interest are: the Lanczos-extension window (by Cappellini) having 1-D expression

$$w_{LC}(t) = \left(\frac{\sin(\pi t/\tau)}{\pi t/\tau} \right)^m \tag{4}$$

being zero for $|t| > \tau$ with m a positive number controlling the shape and the Weber-type window, having the property of giving a minimum value of the uncertainty product suitably modified. The expression of this last window is somewhat complicate: a practical approximation to it has been however defined by Cappellini [1] through a polynomial representation (up to the third order).

For IIR digital -filters (1-D and 2-D) the design must consider also the stability problem, due to the feedback action carried out by the coefficients b(k) or $b(k_1,k_2)$. For 1-D case,techniques of transformation from the continuous domain (as the bilinear transformation) can be used. For 2-D case the design is more complicate and involved, because in the 2-D case it is not possible in general to factorize the z-transform $H(z_1,z_2)$, due to the lack of an appropriate factorization theorem of algebra: some approaches are however available making use of transformation from 1-D to 2-D or division of $H(z_1,z_2)$ in two or four quadrants [1] . An efficient method of this type was defined by Bernabo', Cappellini and Emiliani [1] .

2. MULTIPLE FAST PROCESSING BY MEANS OF DIGITAL FILTERING AND DECIMATION PROCEDURE

Digital filtering is very useful to perform multiple processing in efficient way, with the possibility to make any frequency operation (e.g. low-pass filtering for frequency band-limiting, high-pass or band-pass filtering for extracting some frequency band of interest...).

At first, considering a single sampled signal, it is possible with digital filtering of low-pass or band-pass type to extraxt only the frequency band of interest: a reduction of the sample number can be hence performed to the minimum value required by the sampling theorem [1] .

Multiple fast processing can be performed by using digital filtering as above and sample number reduction; two more interesting situations are: multiple filtering (as band-pass analysis) on a single sampled signal, a required filtering (low-pass, band-pass, high-pass) on several sampled signals.

A system of the second type, processing multiple signals (channels 1...n), with data compression (reduction of the sample number) and adaptive operation is shown - in block-diagram - in Fig. 1. Several sampled signals or data channels are digitally pre-filtered (low-pass or band-pass) in such a way to reduce the data to the minimum value required by the sampling theorem. The filtered data are sent to data compressors (constituted in the simplest form by sample number reduction or decimation, or by prediction--interpolation algorithms, variable length encoding, transformations in other forms [3]): the compressed data are finally sent to output buffers, where the remaining important data are eventually reorganized at constant time intervals to achieve bandwidth compression. The adaptivity is obtained by varying the bandwidth of the digital filters (in general at constant time intervals) by means of a "control circuit" which continuously measures the state of fullness of the output buffers, avoiding the undesirable situations of "overflow" and "underflow" [3] .

A general block-diagram of a system of the first type, performing multiple processing on a single signal, is shown in Fig. 2: the input unit to perform sampling and analog-to-digital (A-D) conversion and the output unit to make digital-to-analog conversion (D-A) and signal reconstruction are also enclosed. The programmer controls the arithmetic unit to perform the different digital filtering operations (of FIR type, using only input data) exchanging data with the memory (containing both the coefficients of digital filtering and eventually output data before to go to final reconstruction block). A multiple processing case of particular interest is represented by the band-pass analysis to be performed on a single input signal. Two methods on this line are described in the following.

A method uses a single digital filter to perform proportional bandwidth band-pass analysis. To illustrate the method, consider the Fig. 3, in which the frequency response of a band-pass digital filter is represented with reference to the maximum frequency of the input signal (ω_M): the band of the digital filter is set between ω/ω_M =1/8 and ω/ω_M = 1/4. In a first filtering the band of the signal between ω/ω_M= 1/8 and ω/ω_M = 1/4 is obtained; hence a time expansion of a factor 2 is applied on the input signal (as shown in Fig. 4) which corresponds to a frequency compression of the sampled signal spectrum of a factor 1/2: by using the same digital filter the band of the signal between ω/ω_M = 1/4 and ω/ω_M = 1/2 is obtained. In a third filtering a time expansion of a factor 4 is applied

on the input signal (as shown in Fig. 4) and by using the same digital filter the band of the signal between $\omega/\omega_M = 1/2$ and $\omega/\omega_M = 1$ is obtained. To perform an analysis on a greater number of proportional bands, it is sufficient to set the band-pass digital filter at a lower frequency band with reference to the maximum frequency of the signal to be analysed.

Another method particularly efficient and fast uses a single low-pass digital filter and a frequency shift operation on the sampled signal spectrum, by multiplying the input data by an exponential factor. Indeed if an angular frequency shift ω_i is performed on the sampled signal spectrum $F(j\omega)$ to produce a spectrum $F(j\omega - j\omega_i)$, then this corresponds to a new sampled signal of the form (with T the sampling interval)

$$f_i(n) = f(n)e^{jn\omega_i T} \tag{5}$$

If ω_M is the signal maximum angular frequency, then angular frequency shifts $\omega_i = i(\omega_M/M)$, $i = 1,\ldots, M-1$, are required to be operated on the signal so as to have M-band analysis by using a single low-pass filter of angular cut-off frequency $\omega_c = \omega_M/M$. It is clear that all the original spectrum frequency bands will be shifted to the low- frequency aband $(0,\omega_M/M)$. In this case the arithmetic unit (of Fig. 2) has to perform complex multiplications on $f_i(n)$ using the coefficients of the single low-pass digital filter [1].

On this line a special method has been proposed by Cappellini [1] [4], using the values $\omega_i = \omega_s/2$ with $\omega_s = 2\pi/T = 2\pi f_s$ as the angular sampling frequency. It is easy now to verify that the relation (5) becomes very simple as

$$f_i(n) = f(n)(-1)^n \tag{6}$$

which implies only an alternative sign change. If moreover $\omega_i = 2\omega_M$ and $\omega_c = \omega_M/2$, the following situation is obtained. Low-pass filtering of the original signal f(n) produces the sampled signal $f_1(n)$ which corresponds to the original spectrum band $(0,\omega_M/2)$, while low-pass filtering of $f(n)(-1)^n$ produces the sampled signal $f_2(n)$ corresponding to the original spectrum band $(\omega_M/2,\omega_i)$, shifted and inverted within the frequency band $(0,\omega_M/2)$. This procedure can be applied again to the signals $f_1(n)$ and $f_2(n)$ by considering only one of two consecutive samples (i.e. involving sample reduction or decimation) so that the preceding relation (6) can be still valid if an angular frequency shift equal to $\omega_M/2 = \omega_s/4$ is used. Then four bands can be obtained. Therefore this method can separate, in a tree structure; the signal spectrum in a rapidly increasing number of bands. In general if r successive operations are performed, then 2^r bands are obtained.

The band-pass analysis resulting from the above procedure presents the following advantages with respect to the normal band-pass filtering:

(1) the band-pass analysis can be obtained by using a unique digital filter without changing its coefficients (depending only, as in FIR filters, on the number of processed samples and on the ratio ω_s/ω_c);
(2) high and constant efficiency of digital filtering;
(3) the speed of analysis is increased because at each step the minimum number of samples connected to the sampling theorem is processed and also because the analysis can be stopped in those parts of the frequency spectrum having zero value or which are of no interest;

(4) due to the sample reduction procedure a compressed representation of each analysed passband is obtained with fewer samples than normally needed, making faster subsequent digital processing operations;
(5) the practical implementation can be relatively simple because the coefficient memory contains only one coefficient set and the operation $f(n)(-1)^n$ is very simple, as shown in the block diagram of Fig. 5.

By performing the r.m.s. evaluation of the different band outputs (in a given time interval T_m) a short-time spectral estimation can be easily obtained through the above technique.

An hardware implementation of the above technique was realized, including the r.m.s. evaluation [1] [5] : an example of processing an audio signal is shown in Fig. 6, reporting the spectral estimation up to a resolution of 16 bands (r = 4).

A software implementation of the same technique was also implemented: an example of processing an audio signal is shown in Fig. 7 (a) with spectral estimation having a resolution up to 32 bands (r = 5). As comparison in Fig. 7 (b) the spectral estimation through the standard Fast Fourier Transform (FFT) is also shown : the good agreement of the two methods in giving the desired result is clearly appearing. The advantages of the presented technique in comparison with FFT are connected to the above considerations (from (1) to (5)) and to the great flexibility in obtaining spectral estimations in the desired time interval. These advantages are particularly relevant when both a band-pass analysis (with output samples corresponding to the different bands) and a spectral estimation are required, because spectral estimation is requiring simple and fast computation (r.m.s. evaluation).

CONCLUSIONS

The described digital processing techniques, using in different ways digital filtering operations, are interesting for high-speed digital acquisition and processing of signals and images (several presented techniques can be easily extended to image processing), in particular for underwater research.

A reduction of the amount of data to the minimum value required by the sampling theorem can be obtained through low-pass or band-pass digital filtering and subsequent decimation. Multiple efficient processing can be performed, particularly convenient - as obtained through the special technique described - when both band-pass analysis and spectral estimation are required, as confirmed by the presented examples of signal processing by means of hardware and software implementations.

REFERENCES

1. CAPPELLINI, V., CONSTANTINIDES, A. G., EMILIANI, P. Digital Filters and Their Applications, Academic Press, London - New York, 1978.
2. HUANG, T. S. Two-dimensional windows, I.E.E.E. Trans. Audio Electroacoustics, vol. AU-20, n. 1, p. 88-89, 1972.
3. CAPPELLINI, V., LOTTI, F., ORI, C., PASQUINI, C., PIERALLI, F. Application of some data compression methods to ESRO satellite telemetry

data, Alta Frequenza, vol. 43, n. 9, p. 673-681, 1974.
4. CAPPELLINI, V., EMILIANI, P. L. A special-purpose on-line processor for band-pass analysis, I.E.E.E. Trans. Audio Electroacoustics, vol. AU-18, p. 188-194, 1970.
5. CAPPELLINI, V., EMILIANI, P. L., GABBANINI, A. A digital processor for on-line band-pass analysis, spectral estimation and signal recognizing, Alta Frequenza, vol. 43, n. 5, p. 307-309, 1974.

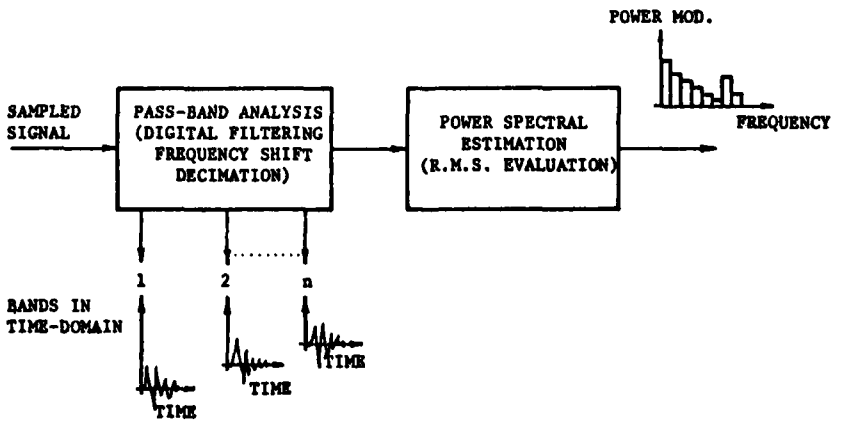
DISCUSSION

T.E. Curtis Is this decimation technique equivalent to a Radix N DFT? Does it require a comparable number of operations (i.e. multiply/accumulate cycle)?

V. Cappellini The decimation technique is near to a Radix N DFT. The overall procedure presented — consisting of a low-pass digital filter, frequency shift of the sampled signal spectrum, and decimation — is quite different, because here the first processing result is a time-domain sampled signal for each pass band from which, if required, a spectral estimation can be obtained (in particular, power spectral estimation through r.m.s. evaluation of each band output).

The number of operations involved can be comparable, if the number of coefficients of the FIR low-pass digital filter is suitably selected. (If the low-pass digital filter is selected with high efficiency — low ripple in band, high cutoff frequency, and low fluctuation out of band — the number of operations here can be higher but a more efficient band separation can be obtained.)

The described technique is of particular interest when a time-domain pass-band analysis in compressed form (that is, each signal in a band is represented through the minimum number of samples) is initially required and also a power spectral estimation is to be performed.



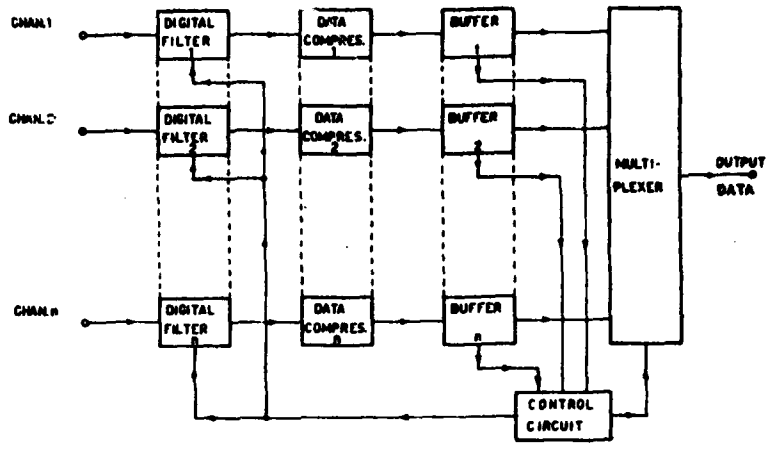


FIG. 1 BLOCK DIAGRAM OF A MULTIPLE PROCESSING SYSTEM WITH DIGITAL FILTERING AND DATA COMPRESSION OPERATIONS UNDER ADAPTIVE CONTROL BY A CONTROL CIRCUIT.

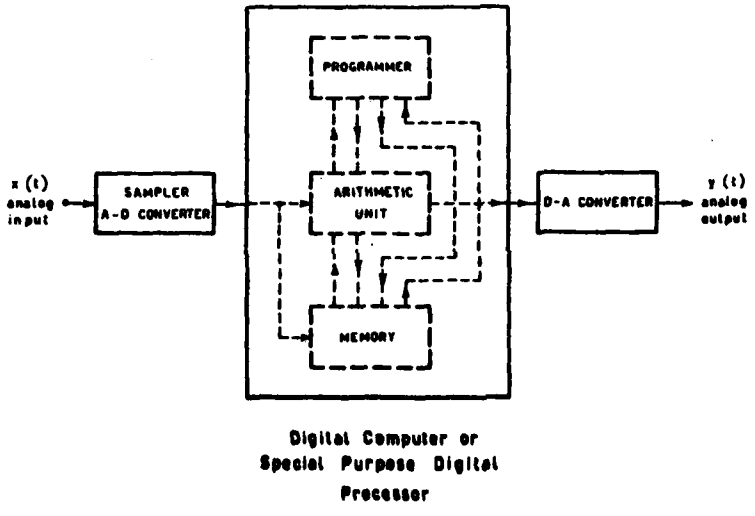


FIG. 2 BLOCK DIAGRAM OF SYSTEM PERFORMING MULTIPLE PROCESSING ON A SINGLE SIGNAL WITH INPUT-OUTPUT UNITS.

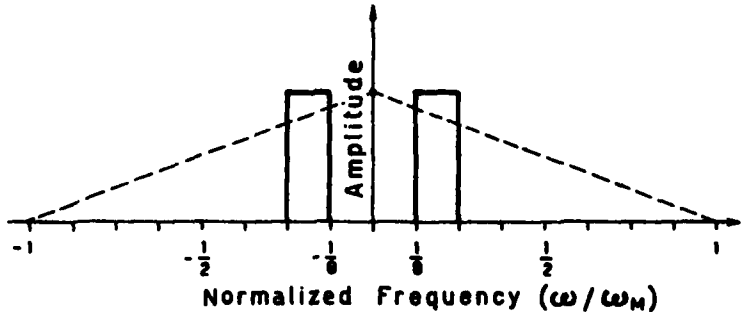


FIG. 3 FREQUENCY RESPONSE OF A BAND-PASS DIGITAL FILTER FOR PROPORTIONAL BANDWIDTH BAND-PASS ANALYSIS IN RELATION TO THE MAXIMUM FREQUENCY OF THE SAMPLED SIGNAL SPECTRUM ω_M.

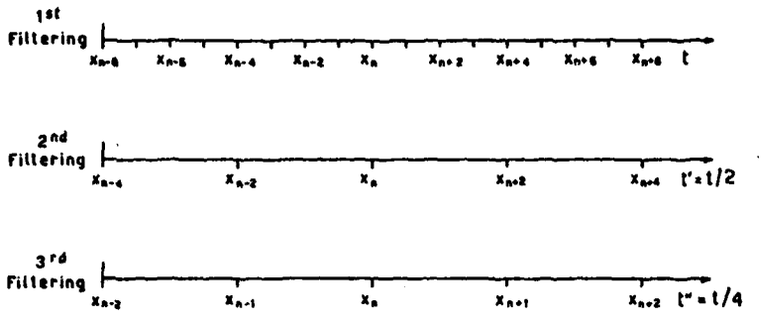


FIG. 4 SIGNAL SAMPLE POSITION IN TIME FOR PROPORTIONAL BANDWIDTH BAND-PASS ANALYSIS (TIME EXPANSION IN 2ND AND 3RD FILTERING)

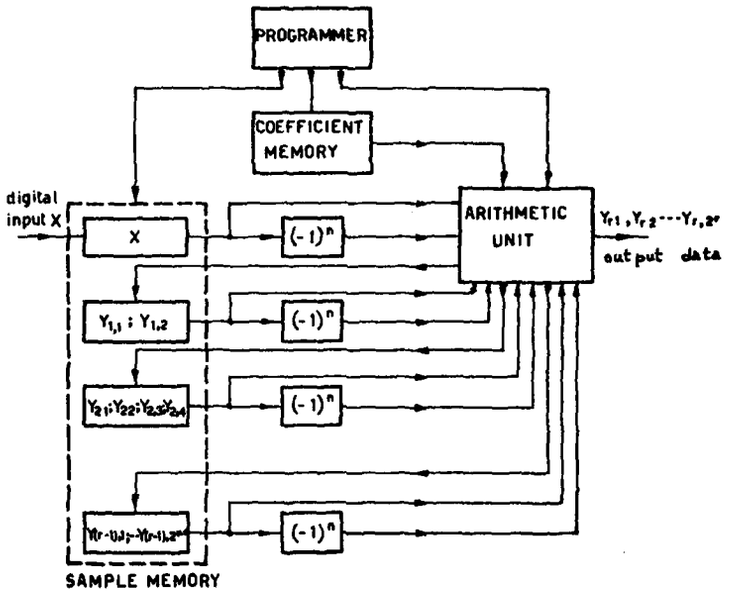


FIG. 5 BLOCK DIAGRAM OF HARDWARE IMPLEMENTATION OF THE DIGITAL PROCESSORS PERFORMING BAND-PASS ANALYSIS (CONSTANT BANDWIDTH) BY USING A SINGLE LOW-PASS DIGITAL FILTER AND A FREQUENCY SHIFT PROCEDURE OF THE SAMPLED SIGNAL SPECTRUM WITH DECIMATION

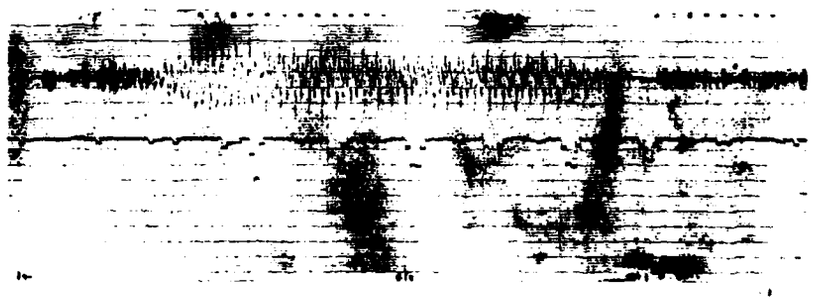


FIG. 6 EXAMPLE OF ON-LINE BAND-PASS ANALYSIS WITH SPECTRAL ESTIMATION ON AUDIO SIGNAL ("GIO") BY MEANS OF THE HARDWARE PROCESSOR; AT TOP - INPUT ANALOG SIGNAL; AT BOTTOM - SUCCESSIVE SPECTRAL ESTIMATIONS REPRESENTED BY 16 r.m.s. VALUES, ONE FOR EACH BAND, IN THE FREQUENCY RANGE 0-4 kHz.

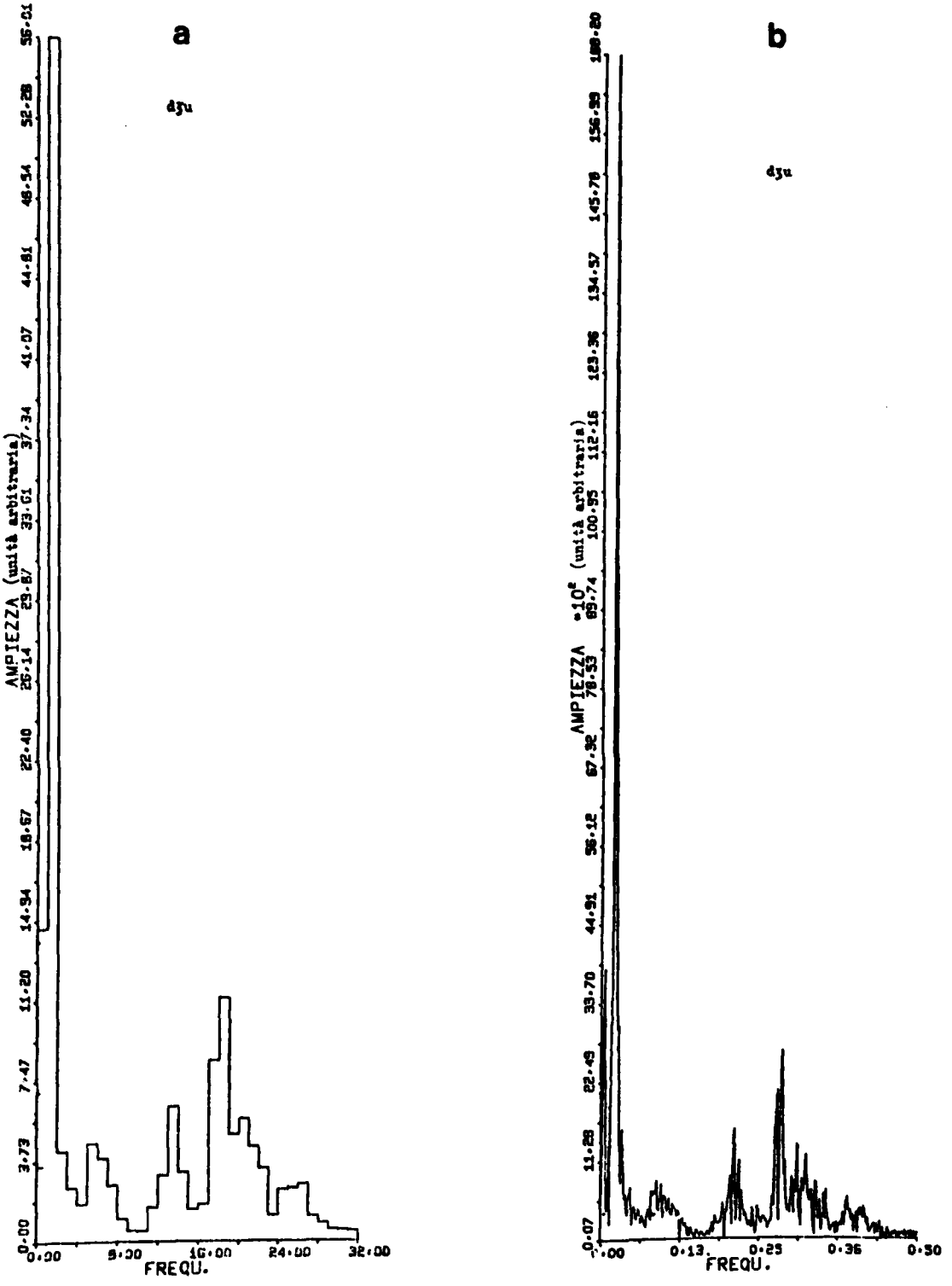


FIG. 7 EXAMPLE OF SOFTWARE IMPLEMENTATION OF THE BAND-PASS ANALYSIS AND SPECTRAL ESTIMATION METHOD AS IN FIG. 5: (a) SPECTRAL ESTIMATION (REPRESENTED BY r.m.s. VALUES) WITH 32 BANDS OF THE AUDIO SIGNAL ("DZU"); (b) SPECTRAL ESTIMATION OF THE SAME AUDIO SIGNAL BY MEANS OF THE STANDARD FFT WITH 512 POINTS.

ARRAYS AND ARRAY PROCESSORS - FUTURE REAL TIME APPLICATIONS IN OCEANOGRAPHY AND SONAR

by

Prof. Arthur B. Baggeroer
Depts. of Ocean And Electrical Engineering
Massachusetts Institute of Technology
Cambridge, Mass. USA 02139

ABSTRACT In the last several years oceanographic applications of arrays have increased substantially. The spatial resolution and redundancy provided by arrays have been necessary in the interpretation of signal propagation in complex environments. While arrays have also been employed extensively in sonar systems, the demands of higher resolution and capabilities of digital signal processing algorithms are still further increasing their use. With is increased use of arrays in oceanography and sonar, the attendent demands of processing the data have also risen. Often, this must be done in real time or at least on site, while the array is still deployed.

Historically the needs of geophysical exploration for oil were a prime mover in the development of array processors. This was principally motivated by the use of large arrays, now with up to 500 channels operating at kilohertz bandwidths, for seismic reflection studies. Fortunately, there are many parallels between geophysical exploration for oil and oceanography and sonar, so some inferences about future applications of arrays and array processors may be made. Here we examine some of these potential applications, particularly those areas in geophysical exploration related to methods of beam forming, frequency wave function estimation, propagation analysis, and remote sensing. We find that arrays and array processors are integral components making possible studies that could not be reasonably accomplished without them.

INTRODUCTION

Much of oceanography and sonar essentially concerns mapping. In oceanography we map the seabed bathymetry, the seismic strata, surface and internal waves and acoustical propagation and reverberation. In sonar we map targets which may be ships

at audio frequencies to medical images at ultrasonic frequencies. The spatial information required to create these maps is acquired by arrays of sensors. These arrays may exist physically as a collection of sensors,or they may be synthesized from repeated observations by a moving sensor. The important point is that arrays are used and all the data required must be processed to convert the information into a map, chart, or image depending upon one's particular application.

In the last several years the technologies for deploying arrays, recording the data, and processing it have increased dramatically. Low power, integrated circuit technology has advanced sensor capabilities; high speed digital equipment has led to recording systems with high dynamic range over wide bandwidths, and the flexibility and speed of programmable array processors have led to rapid, often real-time results and analyses. The full impact of this array technology upon oceanography and sonar has yet to be felt. In fact, the advant of this technology is driving the course of future experiments.

In the sequel we examine future uses of arrays and array processors in oceanography and sonar. While the space available here requires brevity, it is hoped that it is indicative of the range of applications. Moreover, we have not attempted to cite references because of the immense diversity of the field. We draw heavily upon the work that has been done in the geophysical exploration for oil where maps of the seismic strata are the most important data in placing wells. If the experience in geophysical exploration is indicative, then we may anticipate that arrays will strongly influence future oceanographic research and sonar systems.

The discussion of the use of arrays in oceanography focuses on marine geophysics. While other areas in oceanography may also employ arrays, this area will probably dominate the usage for the foreseeable future. Marine geophysics primarily concerns the seabed--its present structure and its evolution. Array applications usually involve the analysis and synthesis of how low frequency sound propagates through the seismic strata. Closely related to marine geophysics is ocean acoustics which focuses upon how sound propagates within the water. Array applications typically involve hydrophones for measuring directional spectra, scattering and multipath phenomena, and propagation analyses.

1 MARINE GEOPHYSICAL EXPLORATION

Marine geophysics as applied to exploring for sources of oil can

claim much of the credit for stimulating the evolutions of arrays and array processors. It is only now with the dramatic reduction in costs brought about by integrated circuits that these technologies are being employed more broadly in oceanography. Marine geophysical exploration uses large, multichannel towed arrays extensively for seismic reflection surveys. Oceanographers are now using them for both reflection and refraction studies. The geometries for these studies are illustrated in Figure 1.

In a seismic reflection operation,an impulsive acoustic source, e.g. an explosive or high pressure airgun,generates signals that propagate into the seismic strata and reflect off the strata interfaces where there are changes in the acoustic impedance of the seabed. This is repeated at closely spaced intervals as the source and array are towed along a "seismic line." Depending upon the equipment and seabed structure, signals at depths up to the Mohorovic discontinuity (15km) can be detected. In a refraction operation an impulsive acoustic source is also used. The separation between the acoustic source and the receiver array, however, is increased to a range that is beyond the critical angles for the seismic strata. At these ranges, interface or head waves are excited. It is fairly common to excite head waves at the Mohorovic discontinuity and operate at source receiver separations in excess of 100km using high charge weights.

The reflected or refracted signals are sensed by hydrophones in a towed array. Electronics on the recording ship perform the necessary signal conditioning and are converted to digital data. Presently there exist towed multichannel arrays with 500 separate channels, up to 10 km in length, and digital recording dynamic range of 120 dB and bandwidths in excess of 1 kHz[6]. This corresponds to recording data rates of 10 M bits/sec. These data (are recorded on magnetic tape for subsequent analysis) or processed in real time, which is becoming increasingly common.

2 ARRAY PROCESSING FOR GEOPHYSICAL EXPLORATION

The array technology for seismic reflection and refraction studies is advancing very rapidly--far faster than comparable sonar efforts. Moreover, the methods of data analysis and the array processing capabilities for implementing them in marine geophysical exploration are also well advanced. The overall significance of both these observations is that we may make some valuable inferences about future applications in oceanography and sonar from this area. The data analysis for mapping the seismic strata in both reflection and refraction involves stack-

ing a beamforming operation, and velocity analysis, a spatial analog of spectral analysis. The essentials of the signal flow are illustrated in Figure 2. Both the stacking and the velocity analysis are heavily dependent upon the capabilities of array processors, especially for real time operations.

2.1 Velocity Analysis

First we examine the velocity analysis operation. It is fundamentally a problem in frequency-wave number fuction estimation. The complicating factor in geophysical exploration is the inhomogeneity of the medium. At best it is only vertically dependent in well-stratified media, while at worst it is very anistropic in regions such as continental shelf breaks and near med-oceanic ridges. Virtually, all the analysis methods are based upon models for horizontally stratified environments. Velocity analysis refer to the methods of spectral analysis that are employed to resolve this horizontal stratification.

The essential concept of a velocity analysis for both reflection and refraction is similar to an angular spectral estimate, or bearing intensity plot. The major difference is the transient character of the signals. In the reflection operation the wavefront curvature of the nearfield environment must be modelled, this leads to spectral estimates where the intensity of the reflected signals are measured as a function of normal incidence travel time which measures the distance to a reflecting horizon and a root mean square velocity which characterizes the wavefront curvature of the near field environment. The highlights of the spectral estimate are used to specify a velocity model for the seismic strata model by inversion methods.

In the refraction operation the spectral intensity as a function of the source to receiver travel time and the horizontal phase velocity across the array is measured. The highlights of this spectra can also be used with model inversion methods to specify a structure for the seismic strata. Figure 3, for example, illustrates a phase velocity spectra for a thinly sedimented seabed.

In either the reflection or refraction operations, the signal processing required to form the velocity spectral estimates is extensive. (In fact, this was one of the prime motivating factors that led the geophysical exploration industry to develop special purpose array processors). For example, a single velocity spectra may require up to 4000 separate spectral analysis with an equivalent two dimensional data base of 64 x 4 samples. Furthermore, the demands for precise model estimates have led to the introduction of velocity spectral estimation methods that are computationally intensive, such as those based upon high resolution spectral estimation algorithms. The important emphasis is that array processors will be extensively used

in oceanography for resolving the spectral structure of signals propagating in the complex environment of the seabed. More particularly, there will be strong emphasis for real time operations so that the important features of an area can be defined and surveyed while the research vessel is in location.

2.2 Stacking and Beamforming

The seismic model from a velocity analysis in reflection operations is used to specify a stacking, or beamforming, operation. The output of this stacking operation is the map of the seismic strata. In its simplest form, the stacking operation consists of determining the travel time delays to a particular depth and then performing travel time dependent, delay and sum, time domain beamforming, which is illustrated in Figure 4. Typically, however, more complicated processing in both the time and frequency domains is also done. The most important processing methods are time varying filtering, deconvolution, velocity filtering, common depth point stacking, and migration. These are illustrated in Figure 5. All can and are done very efficiently using array processors and all are very applicable to beamography and sonar. The properties of the acoustic sources and the frequency selective attenuation properties of the seismic strata as well as the ambient noise require multichannel bandpass filters whose characteristics ideally change with travel time into the strata. Finite impulse response (FIR) and recursive infinite impulse response (IIR) filters with time varying coefficients are commonly used for time domain implementations. In addition, fast Fourier transform (FFT) methods are also used for frequency domain implementations. All of the applications are strongly dependent upon the extensive digital filtering literature in this area. Since both oceanography and sonar often involve frequency selective phenomena, these advances in filtering theory are very relevant to future signal processing.

Deconvolution is a term used to describe a large number of signal processing algorithms that attempt to compress long duration signals into impulsive ones. There are, however, two important applications of deconvolution that are used extensively. In the first, an FIR filter is designed using the data itself which attempts to compress the finite duration impulse response of the source into an impulsive one. Unfortunately, there are several theoretical considerations that indicate that this cannot be done in most practical situations. In the second application one also constructs an FIR filter from the data which attempts to remove the reverberation that is excited by multiple reflections between the sea surface and seabed. There are limitations upon the effectiveness of this application as well. All deconvolution procedures are fundamentally coupled to linear filtering theory, so there is a rich literature supporting them. More importantly, deconvolution can be shown to consist of a whitening operation which is a fundamental operation in spectral analysis, dereverberation and optimal filtering method. With the appearance of array processors multichannel

deconvolution filters are being investigated extensively in marine geophysical exploration as well as in many other fields.

Velocity filtering is intrincically a spatial filter that exploits the properties of the multidimensional Fourier transforms of the propagating signals. The essential concept is that the wave equation governing a propagating signal constrains the allowable domain for its temporal frequency and spatial wave number. Signals that may be present that propagate via different wave equations, e.g. surface waves and noise may be rejected to the extent that they do not overlap the allowable frequency wave number domain of the desired signal. The array geometry, more particularly its finite extent and the spatial separation of the sensors, limits one's ability to do this exactly. As a result, there is a large amount of literature on array processing algorithms that is now accumulating in the digital signal processing literature journals. Much of this concerns the design of velocity filters for realistic array geometries, which is very relevant to future beamforming methods in both oceanography and sonar. Implicit in all this work are the tradeoffs vis a vis array processing capability to implement these filters.

Common depth point (CDP) stacking is the basic beamforming operation in marine geophysical exploration. It's output is the map, or time section, of the seismic strata. An example of one is illustrated in Figure 6. In generating a CDP stack, a sequence of reflections from successive sources impulses are used to synthesize an array. By proper spacing which is created by appropriate timing of the source impulses as the ship moves through the water the correct geometry for focusing the synthesized array can be created. The CDP method has proven to be surprisingly robust, and several attempts to improve upon it using adaptive filtering have not been particular successful. It is, however, computationally demanding. For example, 1km of a typical seismic section requires 10^5 CDP points, or beam outputs, involving 10^8 operations. When combined with the computational burdens of the signal processing previously discussed that occur before the CDP operation, it is easy to appreciate the demands for both high speed array processors and efficient computational algorithms. Nevertheless, several real time systems for processing marine geophysical data at sea now exist. The important point is that both the hardware and software for high speed implementations of the CDP is very applicable to mapping and beamforming operations now done in oceanography and sonar.

Migration is the final step in the signal processing for marine geophysical data. It is an active field of research which uses array processors extensively. Fundamentally, it converts map of seismic section versus travel time into a map versus true depth. In most respects the travel time and depth maps are quite similar; however, the near field array geometry introduces

effects such as diffractions which distort the time section from a true indication of the reflector depths. This is easy to perceive if one notes that the reflection sequence from a point target generates a hyperbolic time section as one passes accross the point of closest approach. Most of the current research on migration is strongly coupled to partial differential equations where the recorded multichannel array data forms the boundary conditions. Generally the equations are factored wave equations which result in parabolic partial differental equations. While equations of this type are directly relevant to calculating range dependent sonar propagation conditions, the more important perspective is that the migration is an array processing method that directly incorporates the propagation dynamics and their effects upon the signals while mapping the true structure of the seismic strata. This is very valuable since it constrains the signal processing in a way that is fundamentally coupled to the physics. Continuing further,wave equation models for signal propagation occur in all facets of ocean ography, so migration methods are applicable to a diverse number of problems. The major difficulty in employing migration methods is that they are computationally demanding. The capabilities of an array processor are needed in order to solve the partial differential equations that are intrinsic to the methods.

2.3 Synthesis of Multichannel Data

Signal propagation in the seabed is complex, and it requires complex models. One encounters many types of waves which use sophisticated mathematical methods, particularly the complex variable theory, to obtain closed form results in even the simplest of situations. This has led investigators to revert to the original partial differential equations for predicting the behavior of signals propagating in the seabed.

The methods that have evolved can be grossly separated into time domain and frequency domain procedures. In the time domain methods one numerically determines the impulse or step response by integrating the wave equations subject to the appropriate boundary conditions at the strata interfaces. Performing this is computationally very demanding.

In the frequency domain, procedures one determines the solution of the wave equation with harmonic excitation. There are a number of techniques available for doing this. The most extensively used involves characterizing the medium into incremental layers with plane wave reflection and transmission scattering matrices for the downgoing and upgoing waves across these layers. One must then integrate overall the plane wave components in the source radiation pattern to obtain the complete response at a particular temporal frequency. In an alternative approach, one separates the variables of the inhomogeneous wave equation,

solves this equation numerically, and then sums over the separated variable, the horizontal wave number to obtain the complete solution at any particular temporal frequency. In both methods fast Fourier transform methods are used extensively. The plane waves in the spatial dimension lead naturally to discrete Fourier representation when they are sampled spatially and the temporal Fourier representation is intrinsic for a time invariant system. These methods are being used extensively now in geophysical research. The principal motivation is that the traditional use of arrival time for the various waves encountered does not lead to unique models. As a result, one is now using amphlitude dependences to further constrain the model. These methods also help in determining the importance of mode conversion such as the compressional to shear among the various waves. This is particularly difficult in refraction analysis when one can only observe compressional waves on a hydrophone.

All these methods are very demanding computationally. Array processors are really the only possible option for the forseeable future in performing the numerical analysis required by them in a reasonable amount of time.

SUMMARY

We have given a very brief overview of how arrays and array processors are used for geophysical exploration. These methods have been used extensively in the search for oil. With the increased use of arrays in oceanography and sonar, all of them are very applicable to future research, especially when one demands real time operations.

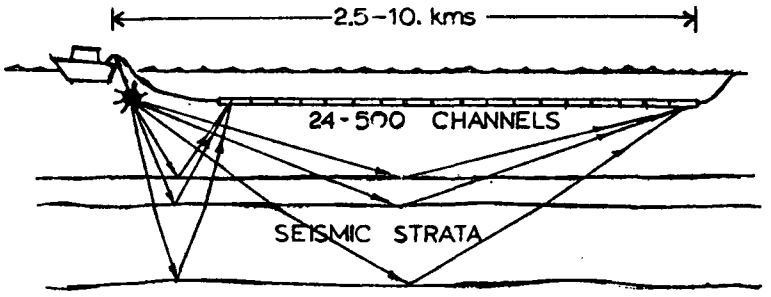


FIG. 1a SEISMIC REFLECTION

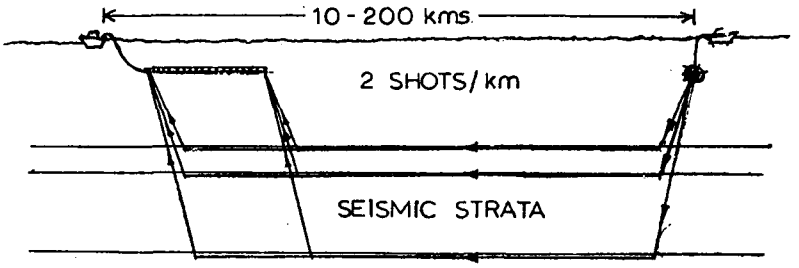


FIG. 1b SEISMIC REFRACTION

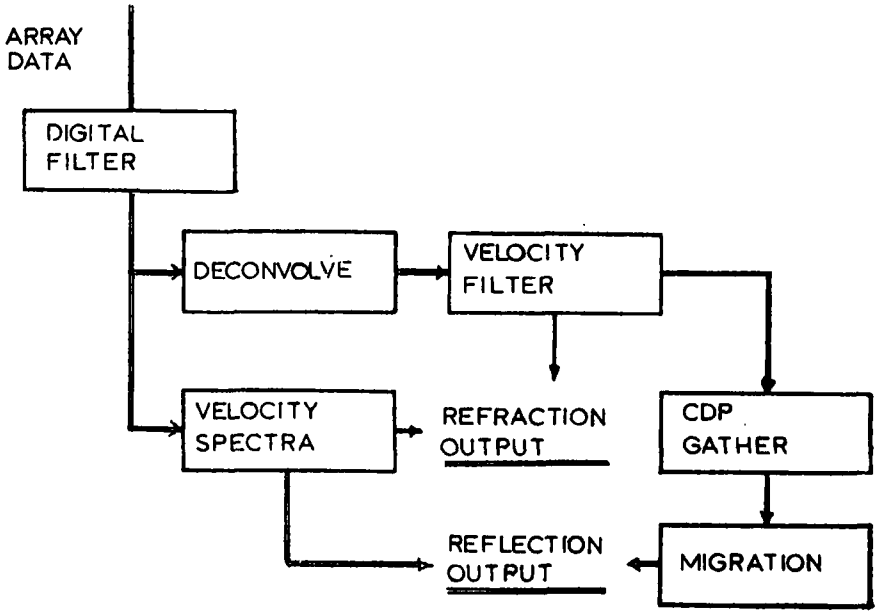


FIG. 2 GEOPHYSICAL EXPLORATION SIGNAL FLOW

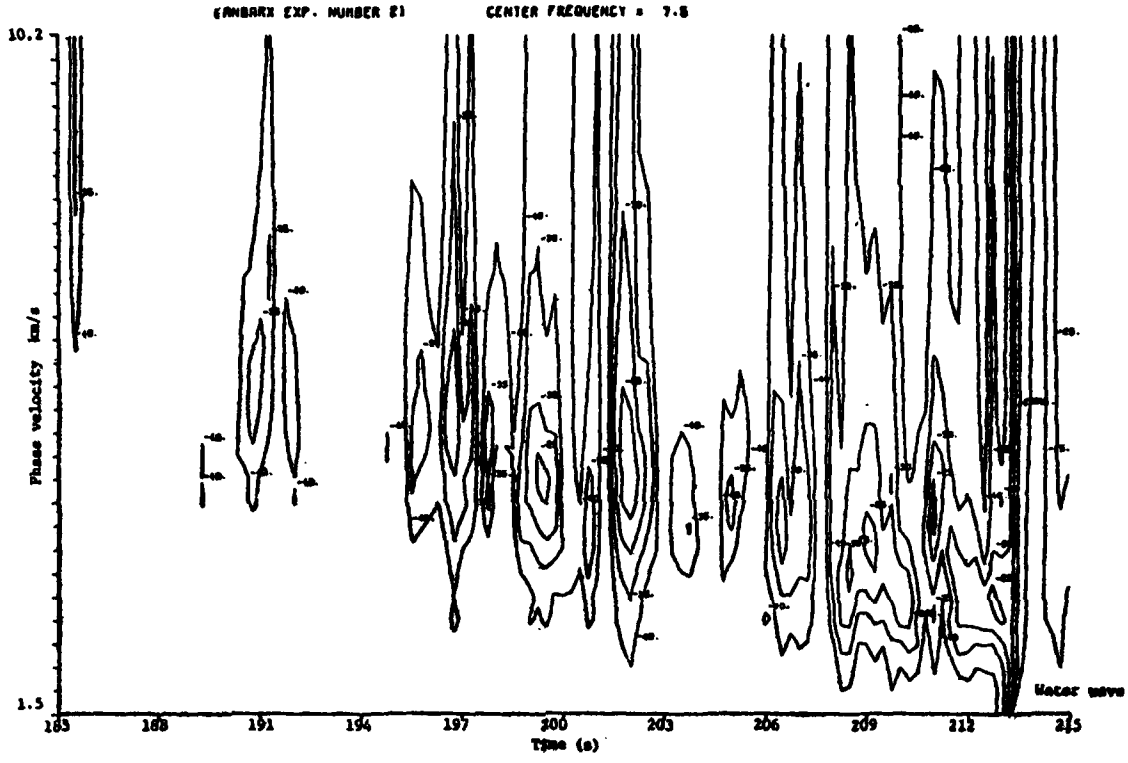


FIG. 3 PHASE VELOCITY SPECTRAL ESTIMATE FOR REFRACTED SEISMIC WAVES

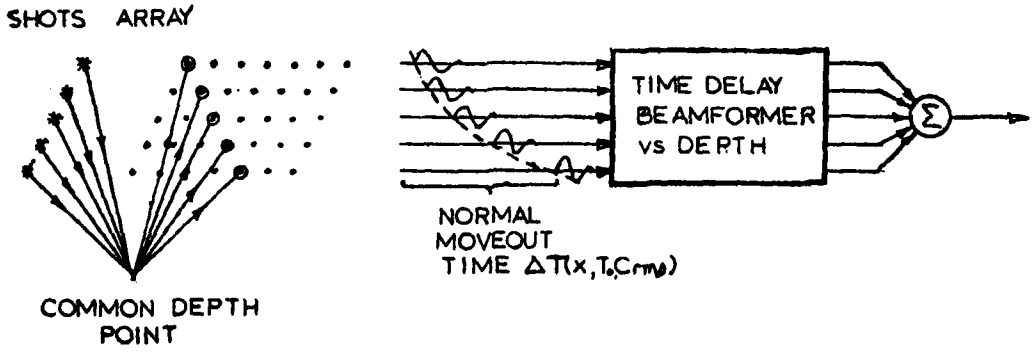


FIG. 4 COMMON DEPTH POINT STACKING

FIG. 5a DECONVOLUTION OPERATION

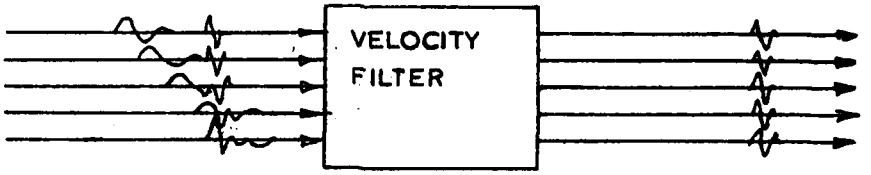


FIG. 5b VELOCITY FILTERING

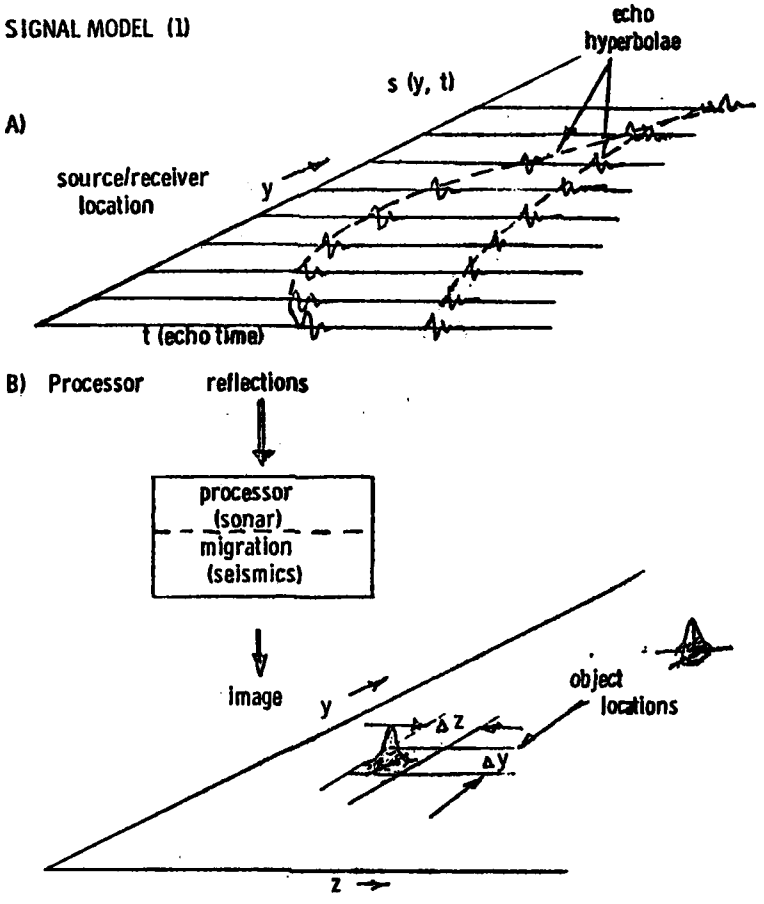
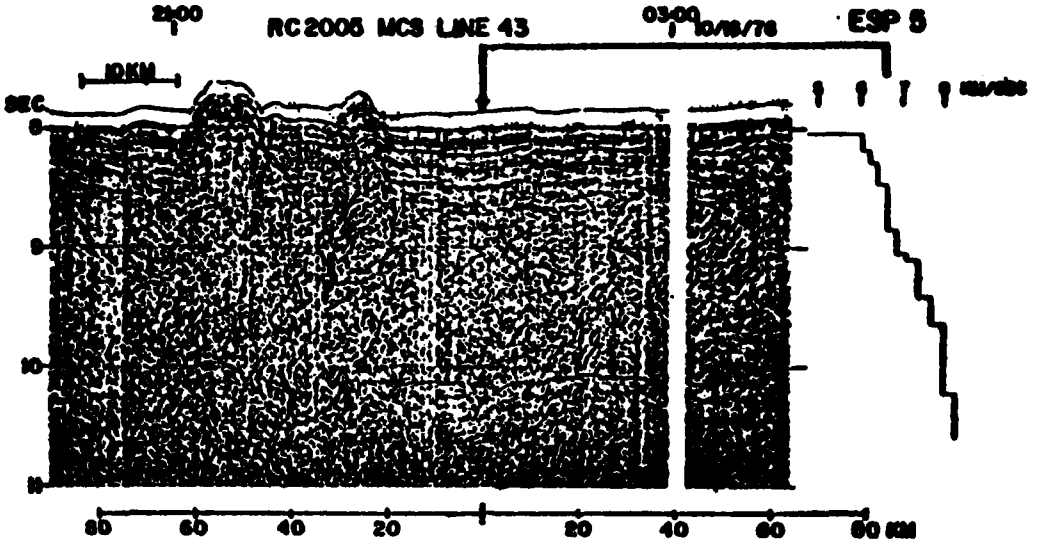


FIG. 5c MIGRATION OPERATIONS



FROM P. STOFFA AND P. BUHL, LAMONT-DOHERTY GEOLOGICAL OBSERVATORY (1978)

FIG. 6 CDP SEISMIC PROFILE